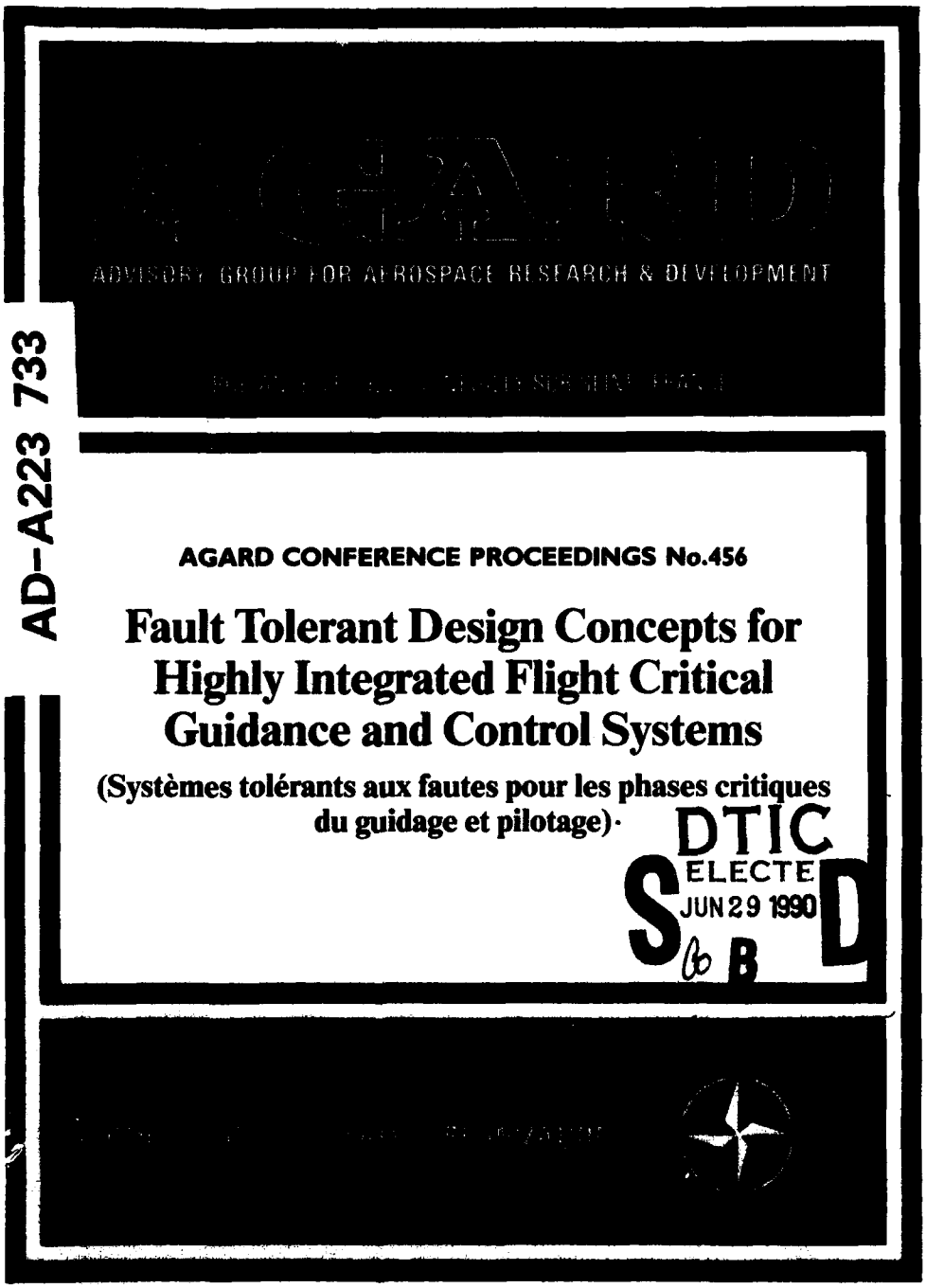
(1)



AGARD-CP-456






AGARD CONFERENCE PROCEEDINGS No.456

# Fault Tolerant Design Concepts for Highly Integrated Flight Critical Guidance and Control Systems

**(Systèmes tolérants aux fautes pour les phases critiques du guidage et pilotage)**







DISTRIBUTION AND AVAILABILITY ON BACK COVER

AGARD-CP-456

NORTH ATLANTIC TREATY ORGANIZATION

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT

(ORGANISATION DU TRAITE DE L'ATLANTIQUE NORD)

AGARD Conference Proceedings No.456

# Fault Tolerant Design Concepts for Highly Integrated Flight Critical Guidance and Control Systems

(Systèmes tolérants aux fautes pour les phases critiques
du guidage et pilotage)

Papers presented at the Guidance and Control Panel 49th Symposium, held at the Ecole Nationale Supérieure de l'Aéronautique et de l'Espace in Toulouse, France, 10th–13th October 1989.

## THE MISSION OF AGARD

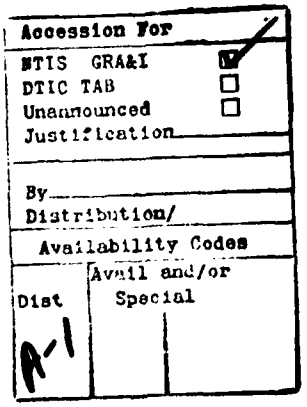
According to its Charter, the mission of AGARD is to bring together the leading personalities of the NATO nations in the fields of science and technology relating to aerospace for the following purposes:

- Recommending effective ways for the member nations to use their research and development capabilities for the common benefit of the NATO community;
- Providing scientific and technical advice and assistance to the Military Committee in the field of aerospace research and development (with particular regard to its military application);
- Continuously stimulating advances in the aerospace sciences relevant to strengthening the common defence posture;
- Improving the co-operation among member nations in aerospace research and development;
- Exchange of scientific and technical information;
- Providing assistance to member nations for the purpose of increasing their scientific and technical potential;
- Rendering scientific and technical assistance, as requested, to other NATO bodies and to member nations in connection with research and development problems in the aerospace field.

The highest authority within AGARD is the National Delegates Board consisting of officially appointed senior representatives from each member nation. The mission of AGARD is carried out through the Panels which are composed of experts appointed by the National Delegates, the Consultant and Exchange Programme and the Aerospace Applications Studies Programme. The results of AGARD work are reported to the member nations and the NATO Authorities through the AGARD series of publications of which this is one.

Participation in AGARD activities is by invitation only and is normally limited to citizens of the NATO nations.




Published April 1990



ISBN 92-835-0552-2

*Printed by Specialised Printing Services Limited*
*40 Chigwell Lane, Loughton, Essex IG10 3TZ*

## THEME

The trend towards highly integrated systems continues to expand at a rapid rate. Recent examples include automated maneuvering attack systems, flight control/fire control coupling, mission sensor management, real-time armament fuzing and propulsion coupling/performance optimization.

The prospect of improved mission effectiveness through integrated systems is a very real and powerful motivation with far reaching implications. Recent advances in microprocessor technology are bringing about fundamental changes in several traditional functional domains. Specifically, systems architecture requirements, partitioning considerations and functional performance parameters take on new meaning in the context of fully integrated flight critical systems. Effective system integration focuses on end-item functional performance using the most efficient mechanization possible. In this regard, system wide consideration of sensing elements, computational elements and command signalling loops are critically important. Crew station design considerations and the pilot's role must also be thoroughly assessed vis-a-vis varying levels of task automation and overall system wide integrity management requirements.

Achieving the full potential of integrated systems is highly dependent upon demonstrating adequate reliability, safety and survivability. Historical evidence indicates that interfacing subsystems can introduce serious compromises in overall system safety and performance. High integrity software is essential. Satisfying stringent flight critical system requirements necessitates innovative fault tolerant design approaches and mechanization schemes. Adding redundancy levels across the full spectrum of system elements is a self-limiting approach based on practical considerations of weight, volume, cost and supportability. Reconfiguration strategies, graceful degradation and aerodynamic redundancy are but a few of the modern concepts currently under development. State estimation techniques in conjunction with artificial intelligence technology also offer potential fault tolerance enhancements. Blending system elements for fully integrated or multi-purpose usage under both nominal and extreme operating conditions, requires an intensive system integration effort to achieve acceptable levels of fault tolerance.

This symposium focused on advanced fault tolerant design concepts and their practical application to integrated flight critical military systems.

* * *

Le tendance vers les systèmes hautement intégrés se developpe rapidement. Des exemples récents concernent les manoeuvres automatiques dans la phase d'attaque, le couplage des systèmes de pilotage automatique et de contrôle des armements, les dispositifs permettant la supervision de la mission, la mise à jour automatique d'armes et l'optimisation globale des performances par inclusion du contrôle de la propulsion.

La perspective d'une amélioration de l'efficacité d'une mission grâce à l'intégration des systèmes est une motivation réelle et puissante avec des conséquences à long terme. Les récents progrès dans le domaine des microprocesseurs apportent des changements fondamentaux dans certains domaines traditionnels. Plus précisément, les exigences de l'architecture des systèmes, la répartition des fonctions et les performances des paramètres fonctionnels prennent un nouveau sens dans le contexte de systèmes hautement intégrés contrôlant les phases critiques de la mission. L'efficacité des systèmes intégrés recherche les performances en bout de chaîne en utilisant la meilleure automatisation: les éléments capteurs, les calculateurs et les informations sur l'état du système conditionnent le succès. La conception des postes de pilotage et les rôles des pilotes doivent être définis avec soin en face des tâches automatisées ainsi que les spécifications de l'ensemble du système largement intégré.

L'aboutissement du potentiel total des systèmes intégrés dépend largement de la démonstration d'une fiabilité, sécurité et survivabilité adéquates. Dans le passé, il est apparu que l'interconnexion de sous-systèmes peut conduire à de sévères compromis sur les performances et la sécurité globales du système. Des logiciels à haute fiabilité sont nécessaires. La satisfaction des contraintes dûes à la phase critique de la mission nécessite des concepts nouveaux dans la tolérance aux fautes et dans les schémas d'architecture et d'automatisation du système. L'adjonction de composants, par redondance et à tous niveaux, est un processus qui a ses propres limites pour des questions de poids, de volume, de coût et de réalisation. Les stratégies de reconfiguration, de dégradation acceptables et de redondance aérodynamique sont quelques uns, parmi la multitude, des concepts couramment utilisés. Les techniques d'estimation de l'état du système liées à celles de la technologie de l'intelligence artificielle offrent également un potentiel de résistance aux fautes. L'interconnexion poussée d'éléments du système pour une intégration totale ou une utilisation polyvalente du système à la fois en conditions nominales et en conditions extrêmes nécessite un effort d'intégration intensif pour atteindre un niveau de tolérance acceptable aux pannes.

Ce symposium s'est intéressé aux concepts avancés de systèmes tolérants aux fautes, à leurs applications aux systèmes intégrés militaires "critiques".

## GUIDANCE AND CONTROL PANEL OFFICERS


Chairman: Ir P.Ph.van den Broek
Department of Aerospace Engineering
Delft University of Technology
Kluyverweg 1
2629 HS Delft
The Netherlands

Deputy Chairman: Professor E.B.Stear
Director, Washington Technology Center
University of Washington
376 Loew Hall – FH10
1013 NE 40th Street
Seattle, WA 98195
United States


## TECHNICAL PROGRAMME COMMITTEE

| | | |
|---|---|---|
| Chairman: | Mr J.K.Ramage | US |
| Members: | Dr M.J.Pelegrin | FR |
| | Mr U.K.Krogmann | GE |
| | Pr J.T.Shepherd | UK |
| | Mr D.E.McIver | US |
| | Dr G.T.Schmidt | US |
| | Pr E.B.Stear | US |

## PANEL EXECUTIVE


Mail from Europe:
Commandant M.Mouhamad, FAF
Executive, GCP
AGARD-OTAN
7 rue Ancelle
F-92200 Neuilly sur Seine
France
Tel. 33 (1) 4738 5780
Telex 610 176F
Fax 33 (1) 4738 5799

Mail from USA and Canada:
AGARD-NATO
Attention: GCP Executive
APO New York 09777


## HOST COORDINATOR


Dr Marc J.Pelegrin
Haut Conseiller
ONERA/CERT
BP 4025
2 avenue Edouard Belin
F-31055 Toulouse
France


## ACKNOWLEDGEMENTS/REMERCIEMENTS


The Panel wishes to express its thanks to the French National Delegates to AGARD for the invitation to hold this meeting in their country and for the facilities and personnel which make the meeting possible.

Le Panel tient à remercier les Délégués Nationaux de la France près l'AGARD de leur invitation à tenir cette réunion dans leur pays et de la mise à disposition de personnel et des installations nécessaires.

CONTENTS

# KEYNOTE ADDRESS

by

**Général François Maurin**
Member of French Conseil d'Etat and Former Chief of Staff of the French Armed Forces

## I INTRODUCTION

Thank you for inviting me to give the keynote speech to this 49th GCP symposium on "Fault Tolerant Design Concepts for Highly Integrated Flight Critical Guidance and Control Systems". This topic will certainly be one of the major concerns of aeronautics from now until the year 2000.

If NATO air forces are to maintain their superiority over the air vehicles and missiles of our potential adversaries, whose aerodynamic performances and flight envelopes are close to or equal to those of our own aircraft, then the only possible solution is to concentrate our efforts on increasing and improving our flight control and combat aids, and thereby maintain our technology lead in this area.

I am sure you will agree with me when I say that in the coming decades, weapon system reliability will prove vital to the success of missions performed by combat aircrew in an increasingly hostile environment. Numerically inferior, the availability of our air vehicles and the speed of their reconfiguration must be considerably improved.

Our weapon systems will need to be increasingly accurate, safe, reliable, all-weather and computerised, so as to allow the pilot to optimise his threat response and achieve his aims, while at the same time reducing his workload.

In order for this to happen, stress must be placed on the guidance and control functions of the weapon system and on their ability to deal with internal errors and false alarms, not just in computers, but in all components.

Increasing computerisation means that the various system functions are becoming highly critical, and the loss of one of these functions during a crucial flight phase can be catastrophic, resulting in mission failure.

*System* architecture should ensure not only increased performance, but also greater reliability and simple, rapid maintenance.

In this context, the internal organisation of the architecture management of a sophisticated and highly integrated weapon system and its error tolerance is much more complex than for a normal computer, as in addition to its links with on-board computers, system reliability is closely bound up with the design and management of the following:

- the sensors and their interfaces
- operation of the various subassemblies
- data transmission along the network which interlinks them
- reliability of the software and the capabilities of the language chosen.

## II THE TECHNICAL CONSTRAINTS

Having had the honour of commanding the French Air Force Experimental Centre at Mont de Marsan, and having been Chief of Staff of the French Armed Forces in the 1970s, I can measure the progress achieved in your field by developments in French weapon systems, or in systems produced in conjunction with our European partners, in both the civil and military domains.

Sufficient as examples, are the developments and technological advances which have taken us from the first generation of Mirage III's to the Rafale, by way of the Jaguar, and the equally spectacular progress which separates the Transal from the family of Airbus aircraft.

This notwithstanding, there remain a number of permanent features of the question, on which it would be advisable to concentrate our attention, in order to make our weapon systems more efficient by reducing procurement and deployment costs.

From the user's point of view, these features can be summarised as follows:

- a reduction in weight and volume
- reliability and survivability
- a reduction in the amount of connections between subassemblies, which are a frequent source of faults and corrosion
- system versatility, interchangeability and modularity
- redundancy for data transmission systems
- software reliability and adaptability
- tolerance of internal errors and downgraded and rapid reconfiguration mode capability
- ECM and partial destruction withstand capability
- rapid and simple maintenance procedures.

I am sure that you are as familiar as I am with all these different constraints, but I think it is useful to restate them here. You see, experience has taught me just what feats of ingenuity researchers and specialists such as yourselves are capable of producing in order to meet military specifications laid down by users who are not always too bothered about the cost aspect, but are rather more concerned with the success of the mission.

What happens in these cases is that once a solution is found, insurmountable financial problems arise and the project never sees the light of day because of the lack of credits.

## III THE HUMAN CONSTRAINTS

Having dealt briefly with the technical constraints, I now propose to consider the human aspects of the problem. As far as the project and the pre-project are concerned, there must be a continuous dialogue between designers, users, and the multidisciplinary team responsible for coordination.

This well known management rule becomes even more vital when designing a highly integrated guidance and control weapon system. In addition to the processing and management of mission data, it imposes and will continue to impose a series of technological decisions on the equipment manufacturers who design the various sensors and flight controls (and even engine designers), and this, some five years before production of the prototype. They must therefore be informed of the specifications to be met in order to produce data in the form of signals which are easy to process in a highly integrated system, in order to avoid costly and fault provoking interfaces.

The PAVE PILLAR and PAVE PACE programmes have opened up the way to solutions of this type and your work should lead to a consensus of opinion among the various nations of the Atlantic alliance.

I would stress this point, as in the coming decades we shall need to ensure that a continuous dialogue takes place between the designers of highly integrated guidance and control systems and flight control and sensor designers, otherwise our efforts will be frittered away and we shall continue to produce highly complex and costly systems.

For some of you, who are familiar with this problem, the choice of the best possible trade-off at least cost, is not and will never be an easy one. This choice becomes even more complex once a highly integrated system project brings together computer designers, software manufacturers, equipment manufacturers and the designers of future sensors, all around the same table, in contrast to recent practice, in which the various parties involved always acted independently, being content to let the overall weapon system designers deal with the details of the interfaces.

We must create multidisciplinary teams right from the outset of any such project, prepared to see it through right to the end. Quite apart from the current technological advances which are being made, I am convinced that one of the main sources of enhancement for future guidance and control systems lies in this common approach, right from the initial design stage of the project. Thanks to this continuous concerted effort, in the future it will be possible to limit the complexity and sophistication of highly integrated and computerized systems, so that they better reflect their ultimate purpose, which is the success of the mission.

The fact is that we often still produce subassemblies whose sophistication is not justified by the overall system specification. We must avoid "art for art's sake".

## IV FUTURE PROSPECTS

In the near future, advances and innovations in the field of guidance and control will offer a growing number of users a wide range of possibilities, enabling them to accomplish any given mission in different ways, with equal chances of success.

Competition between unmanned vehicles, manned vehicles and completely autonomous missiles will be increasingly open.

I shall consider only manned vehicles, in order to allow for the role of the human operator in the data management loop and for the actions he is required to take, in the shortest possible time, in response to the information relayed to him.

At the present time, fighter pilots or aircrew, are confronted by a multiplicity of information sources, whose variety and dissimilarity tend to divert their attention rather than to concentrate it, and which increase their workload during the most critical phases of the mission. Pilots must constantly create a balance between synthetic data and their visual perception of their operational environment.

It is therefore of prime importance for the design of future guidance and control systems to remember that one of the ultimate aims of the system is to present the pilot with clear and precise, limited and sequential data, which match the requirements of the moment.

One of the tasks which will be required of the designers of highly integrated systems in the near future, and it is a considerable one, will be to design a data display on the instruments panel which may well involve a complete reconfiguration of the cockpit in next generation weapon carriers.

The promise held out by expert systems, artificial intelligence and the three dimensional display of synthetic data should go a long way to solving this problem.

Clearly, all this innovation cannot but add to the sophistication and complexity of the guidance and control systems in future generation weapon systems; but it may also result in appreciable simplification if we are talking about a completely new system and not the adaptation of new technology to an old concept.

With this in mind, it would seem to me to be an opportune moment in my address to summarize the different technologies employed in the subassemblies making up a guidance and control system, in order to point the way to the architecture of a global concept of "mission data management".

This is certainly a major task, and a tricky one. It will require much time and effort, but I think it is within the reach of your Panel, as our programme of work covers most of the points I have just mentioned.

A study of this type should produce recommendations and guidelines enabling the designers of each part of the "puzzle" which a highly integrated guidance and control system represents, to base the design of their projects on a master plan, in all probability using a common language for data transmission. This should, in time, lead to the adoption of a certain standardisation, with a reduction in the complexity and sophistication of the overall system, accompanied by a reduction in cost. This system approach should be developed in our Universities. I know that the Sup Aéro Institute was one of the first, if not the first, to introduce this aspect into its curriculum.

This study of the subassemblies: "fire-control/missile", "navigation" and "control" should enable us to distinguish the share of guidance and control equipment to be retained on the weapon carrier and the share to be incorporated into the missile in order to enable the target to be attained with the greatest possible autonomy.

It is implicit in this approach that only those guidance and control sub-systems necessary for the different phases of the mission (take-off, flight control, target acquisition and landing) would be retained on board the weapon carrier; the rest of the system being incorporated in the missiles. Obviously this type of analysis would need to take into account the assistance provided by next generation navigation systems of the type GPS-NAVSTAR or aircraft such as AWACS or JTIS etc., so as to reduce, or, where possible, simplify the number of on-board sensors carried.

As I pointed out previously, such studies can be successful only if a continuous dialogue is maintained between all parties involved in the global project.

The line is so fine between the concept of "guidance and control" and that of "avionics", that the question of who should be prime contractor for the global system remains unanswered. Perhaps your inter-Panel GCP/AVP symposia could come up with a recommendation.

One of the weak points of the NATO forces, when faced with a potential threat from the Warsaw Pact, being numerical inferiority, it would seem to me of great interest to know to what extent such concepts would allow us to increase the numbers of our "weapon platforms" while reducing their cost, and at the same time maintaining our technological superiority.

## V THE FINANCIAL CONSTRAINTS

Last, but by no means least, come *the financial constraints*, which increasingly affect the military budgets of all the nations of the Alliance. They are particularly sensitive in the field which interests us. A global cost estimate of the sensors and detectors which combine to make up the high performance guidance and control systems carried by today's manned air vehicles shows that they represent nearly 40% of the total cost of the air vehicle.

In the face of such financial constraints and of the sheer volume of studies which need to be carried out, it is increasingly apparent that the resources required exceed the possibilities of a single country; the need to combine our efforts within the Alliance on promising new technologies is now urgent.

The current negotiations regarding the reduction of conventional weapons and short range strategic missiles, which will eventually lead to numerical parity between East/West weapon systems outside their countries of origin (USA—USSR), will make this kind of cooperation even more necessary.

We shall not only have to cooperate on individual weapon system projects, but also turn our attention to the broad range of weapon systems to be deployed in order to maintain our effectiveness.

We must replace the idea of standing firm in the face of a numerically superior enemy by the idea of adaptation of our forces to the neutralisation of targets both on the battlefield and in depth. This will probably lead us to review our conventional air weapons and, as a result, the guidance and control concepts of future generation weapon systems.

However utopian it may be to think that international industrial competition in this high-tech sector will disappear in the short term, I think it is realistic, in view of what is at stake, which is nothing less than the preservation of the freedom of the western world, that we should combine our research and design efforts so as to produce a number of common core programmes, methods and a certain interchangeability between the various weapon systems, while at the same time allowing each nation the freedom to develop its weapons industry and the choice as to whether or not to cooperate on common aerospace projects.

It is all too often regrettable to see several different nations of the Alliance exhausting their individual financial resources on R&D work on the same technological problem, only to find that the answer to the problem is discovered by each country practically at the same time, with a few rare exceptions, and that the final products have more or less the same performance and use the same innovations.

In many cases, given clearly defined and approved aims, better cooperation between the nations of the Alliance from the outset of the project would have produced the same result at less cost, and certainly a lot faster, without compromising industrial expansion in our respective countries.

Whereas in the past, the nations of the Alliance may have been able to afford this kind of luxury, the major technological challenges in Aerospace in the coming decades, such as SDI, the supersonic transport plane, are of a nature which prohibits such action.

If we were to continue in this egotistical way, then many of these designs would remain at the "drawing board" stage.

Our potential adversaries, who are more pragmatic, would derive benefit from our difficulties, and in addition to their numerical superiority from the onset of any engagement, would be able to match us from the technology point of view.

Any shortfall in programme credits or extension of the time frame for weapon carrier manufacture would mean that NATO would lose the credibility it still has on the conventional weapons side, and which I would qualify as "conventional tactical and battle skills deterrence", which would be detrimental to the overall strategy of the countries of the Alliance.

## VI CONCLUSIONS

AGARD is without doubt the most suitable body to carry out such a study, and at the same time convince your respective authorities of the benefits of concerted cooperation, and within AGARD, GCP is incontestably the most suitable Panel.

You are, in fact, practically the only forum, if not the only group of high level experts, free of governmental constraints, who can devote themselves entirely to innovative research, the formulation of pertinent recommendations and the maintenance of a continuous dialogue between all the experts in the various scientific disciplines which combine to form the basis of the future guidance and control systems to be fitted to our future weapon systems.

## EXPOSE DE L'ESSENTIEL DU SUJET

par

**Général François Maurin**
Membre de Conseil d'Etat français et ancien chef d'Etat-major des armées

## II INTRODUCTION

Merci de m'avoir invité pour prononcer le discours d'ouverture de votre 49ème Symposium de la Commission Guidage et Pilotage dont le thème "Les concepts en matière de tolérance aux pannes pour des systèmes critiques hautement intégrés de Guidage de Pilotage" est certainement un des points majeurs que l'aéronautique de l'horizon 2000 devra surmonter.

Pour maintenir aux Forces Aériennes de l'OTAN leur supériorité face aux vecteurs aériens et aux missiles de nos adversaires potentiels dont les performances aérodynamiques et les domaines de vol se rapprochent ou égalent celles de vos propres vecteurs, la seule issue possible est de concentrer nos efforts sur l'accroissement des aides au pilotage et au combat de nos propres forces et de maintenir dans ce domaine notre avance technologique.

Pour permettre aux équipages et en particulier aux pilotes des avions de combat d'accomplir leurs missions dans un environnement de plus en plus hostile, il est clair et vous en conviendrez avec moi, que la fiabilité des systèmes d'arme est un enjeu capital pour les prochaines décennies. Inférieur en nombre, la disponibilité de nos vecteurs et la rapidité de leur remise en oeuvre doivent être considérablement accrues.

Nos systèmes d'arme doivent et devront être de plus en plus précis, sûrs, fiables, tout temps et automatisés afin de permettre au pilote d'adapter la meilleure réponse à la menace et d'atteindre son objectif tout en diminuant sa charge de travail.

Pour atteindre cet objectif, l'accent doit être mis sur les fonctions de pilotage et de contrôle du système d'arme et sur leur capacité à surmonter les erreurs internes et les fausses alarmes non seulement dans le ordinateurs mais aussi dans tous les composants.

Les diverses fonctions du système devenant très critiques au fur et à mesure que l'automatisation est plus poussée, la perte de l'une de ces fonctions peut être catastrophique dans une phase cruciale du vol et conduire à l'échec de la mission.

L'architecture du système doit non seulement permettre un accroissement des performances mais aussi sa fiabilité et une maintenance simple et rapide.

A ce titre, l'organisation interne de la gestion de l'architecture d'un système d'arme évolué et hautement intégré ainsi que sa tolérance aux erreurs est beaucoup plus complexe que pour un ordinateur proprement dit, car en plus des ordinateurs embarqués, la fiabilité du système dépend aussi étroitement de la conception et de la gestion:

- des capteurs et de leurs interfaces,
- de la mise en oeuvre des divers sous-ensembles,
- de la transmission des données dans le maillage qui les relie entre eux,
- de la fiabilité des logiciels et de la capacité du langage adopté.

## II LES CONTRAINTES TECHNIQUES

Ayant eu l'honneur de commander le Centre d'Expérimentation de l'Armée de l'Air à Mont de Marsan et d'avoir été Chef d'Etat-major des Armées dans les années 70, je peux mesurer les progrès accomplis dans le domaine qui vous préoccupe au travers de l'évolution des systèmes d'arme français ou réalisés en coopération avec nos partenaires européens et ceci aussi bien dans l'aéronautique militaire que civile.

Je ne citerai comme exemple que l'évolution et les réalisations technologiques qui nous ont conduit de la première génération des Mirages III au Rafale en passant par le Jaguar et les progrès tout aussi spectaculaires réalisés entre le Transal et la famille des Airbus.

Il n'en demeure pas moins qu'un certain nombre de constantes demeurent sur lesquelles nous devons concentrer nos efforts afin d'obtenir une meilleure rentabilité des systèmes d'arme pour diminuer les coûts d'achat et de mise en oeuvre.

En me plaçant du point de vue de l'utilisateur, ces constantes peuvent se résumer ainsi:

- *allègement des poids et diminution des volumes,*
- fiabilité et survivabilité,
- diminution des connections entre sous-ensembles qui sont la source de nombreuses pannes et de corrosion,
- versatilité, interopérabilité et modularité des systèmes,
- redondance des systèmes de transmission des données,

— fiabilité et adaptabilité des logiciels,
— tolérance aux erreurs internes et possibilité de travailler en mode dégradé et reconfiguration rapide,
— résistance aux contre-mesures électroniques et à des destructions partielles,
— maintenance rapide et peu onéreuse.

Ces diverses contraintes, vous les connaissez aussi bien que moi mais il m'a semblé opportun de les rappeler car, par expérience, je connais les efforts d'ingéniosité faits par les chercheurs et les spécialistes que vous êtes pour atteindre les spécifications opérationnelles militaires qui vous sont fixées, dans un premier temps, par les utilisateurs qui, bien souvent, les proposent sans trop se soucier des coûts, préoccupés qu'ils sont par la réussite de la mission.

La solution, une fois trouvée, se trouve alors confrontée à des problèmes financiers insurmontables et ne peut voir le jour par manque de crédits.

## III LES CONTRAINTES HUMAINES

Après avoir résumé brièvement les contraintes techniques, je suis amené tout naturellement à vous présenter d'autres contraintes: les contraintes humaines qui doivent être prises en considération. Au niveau de l'avant-projet et du projet: la nécessité d'un dialogue permanent entre les concepteurs, les utilisateurs et l'équipe pluridisciplinaire chargée de coopérer pour sa réalisation.

Cette règle de management bien connue de tous est encore plus impérative lors de la conception d'un système d'armes hautement intégré de Guidage et de Pilotage et de Contrôle.

En plus du traitement et de la gestion des informations nécessaires à la réussite de la mission, elle impose et imposera aux équipementiers qui conçoivent les divers senseurs, les commandes de vol (jusqu'aux motoristes y compris) des choix technologiques et ceci au moins cinq ans avant la réalisation du prototype. Ils doivent donc être informés des spécifications à satisfaire pour produire des informations sous forme de signaux faciles à traiter dans un système hautement intégré afin d'éviter des interfaces coûteux et sources de pannes.

Les programmes "PAVE PILLAR" et "PAVE PACE" ouvrent la voie à cette recherche de solutions et vos travaux devraient pouvoir conduire à un concensus au sein des divers pays de l'Alliance Atlantique.

Je me permets d'insister sur ce point car pour les prochaines decennies il nous faudra veiller à nouer en permanence ce dialogue entre les concepteurs de systèmes hautement intégrés de guidage et de pilotage et les concepteurs de senseurs et de commandes de vol sinon nous disperserons nos efforts et continuerons à produire des systèmes très complexes et coûteux.

Pour certains d'entre vous qui connaissez ce problème, le choix du meilleur compromis possible, à moindre coût, n'est pas et ne sera toujours pas une chose simple à réaliser. Ce choix sera encore plus complexe à faire dès lors que la réalisation d'un projet de système hautement intègré associera autour d'une même table, les concepteurs d'ordinateurs, les producteurs de logiciels, les équipementiers et les constructeurs des senseurs futurs qui, jusque là, faisaient cavaliers seuls et laissaient aux autres concepteurs du système d'arme global le soin de concevoir les interfaces.

Il conviendra en conséquence, dès la genèse du projet de créer des équipes pluridisciplinaires chargées de mener à terme le programme global du vecteur. Indépendamment des progrès technologiques en cours je reste persuadé qu'une des améliorations principales des futurs systèmes de guidage et de pilotage résidera dans cette reflexion en commun dès la phase initiale de la conception du projet. Grâce à cette concertation permanente, il sera possible de limiter dans le futur la complexité et la sophistication des systèmes hautement intégrés et automatisés en fonction sa finalité globale qui est: la réussite de la mission.

En effet, bien souvent encore nous assistons à des réalisations de sous ensembles dont la sophistication n'est pas justifiée le système étant pris dans son ensemble. En la matière il faut éviter de faire de "l'art pour l'art".

## IV LES PERPECTIVES D'AVENIR

Dans un proche avenir, les progrès, les novations prévisibles dans le domaine du guidage du pilotage et du contrôle offriront de plus en plus aux utilisateurs un large éventail de moyens pour accomplir une mission donnée avec les mêmes chances de succès.

La compétition entre les vecteurs non pilotés, les vecteurs pilotés et les missiles entièrement autonomes sera de plus en plus ouverte.

Je ne retiendrai, dans mon propos, que les vecteurs pilotés pour tenir compte de la présence d'un être humain à bord et du rôle qu'il doit jouer dans la boucle de la gestion des données et des actions qu'il doit effectuer, dans un minimum de temps, en fonction des informations qui lui sont présentées.

A l'heure actuelle le pilote de combat ou l'équipage est confronté à de multiples sources d'informations dont la variété et la non homogénéité ont tendance à disperser son attention et à accroître sa charge de travail dans les phases les plus critiques de la mission. Il doit en permanence faire une synthèse entre les informations synthétiques et la perception visuelle de l'environnement dans lequel il évolue.

Il importe donc, dès la conception des systèmes futurs de guidage de pilotage et de contrôle, de garder en mémoire qu'une des finalités du système est de présenter au pilote les informations claires et précises limitées et séquentielles qui correspondent aux nécessités de l'instant.

Un des efforts, et non des moindres, pour le proche avenir que les concepteurs de systèmes hautement intégrés auront à fournir sera de concevoir une présentation des données sur les tableaux de bord qui pourrait aller jusqu'à la reconfiguration du cockpit dans son ensemble des vecteurs de combat des futures générations.

Les perspectives prometteuses des systèmes experts, de l'intelligence artificielle et de la visualisation en trois dimensions des données synthétiques devraient aider à résoudre ce problème.

Il est certain que cette novation peut encore ajouter à la sophistication et à la complexité des systèmes de guidage de pilotage et de contrôle des futures générations des systèmes d'armes; mais elle peut aussi aboutir à des simplifications très appréciables s'il s'agit d'un système entièrement nouveau et non de l'adaptation de technologies nouvelles sur un concept ancien.

A ce titre, le moment me semble venu de faire une synthèse des différentes technologies que les sous ensembles devront constituter un système de guidage de pilotage et de contrôle afin d'amorcer l'ébauche de l'architecture d'un concept global "Management des données nécessaires à la mission".

Cette tâche est certainement très lourde et délicate à conduire et demandera beaucoup d'effort et de temps mais elle me semble à la portée de votre commission car notre programme de travail porte sur la majorité des points que je viens d'évoquer.

Une étude de ce type devrait dégager des recommandations et des directives qui permettraient à chacun des concepteurs d'une partie du "puzzle" que constitue un système de guidage de pilotage et de contrôle hautement intégré (d'un vecteur aérien, piloté ou non), de concevoir son projet suivant un schéma directeur et probablement d'adopter un langage commun pour la transmission des données. Ceci devrait conduire à terme à l'adoption d'une certaine normalisation et réduire la complexité et la sophistication du système global et par voie de conséquence de son coût, approche de système qui doit être développée dès l'Ecole. Je sais que Sup Aéro doit être la première, sinon une des écoles, à introduire cette réflexion dans les cours.

Cette étude au niveau des sous-ensembles: conduite de tir-missile, navigation, pilotage devrait permettre de déterminer la part du guidage et du pilotage à maintenir à bord du "vecteur plateforme" et la part à embarquer dans le "missile" pour atteindre son objectif avec la plus grande autonomie possible.

Cette réflexion sous entend que seuls les sous systèmes du système de guidage de pilotage et de contrôle nécessaires aux différentes phases de la mission (conduite et manoeuvre de l'avion du décollage, à l'accrochage de l'objectif et indispensables au retour et à l'atterrissage) seraient maintenu à bord du vecteur plate forme, l'autre partie du système étant intégrée à bord du ou des missiles. A l'évidence une telle analyse devrait également prendre en considération l'assistance fournie par des systèmes de navigation des futures générations du type GPS-NAVSTAR ou d'avions du type AWACS, GTIS, etc... afin de réduire ou de simplifier, si possible, le nombre des senseurs embarqués.

Comme je l'ai déjà souligné précédemment, de telles études ne peuvent aboutir que si un dialogue permanent est maintenu entre toutes les parties prenantes du Système global.

La frontière est tellement étroite entre le concept de Guidage de Contrôle et de pilotage et le concept "Avionique" du vecteur que la réponse à la question de savoir qui doit être maître d'oeuvre du système global reste posée.

Vos symposia inter Panel GCP—AVP pourront peut-être répondre à cette délicate question.

Un des points faibles des forces de l'OTAN face à une menace potentielle des forces du Pacte de Varsovie restant l'infériorité en nombre, il m'apparait très intéressant de voir dans quelle mesure de tels concepts permettraient d'accroître le nombre de nos vecteurs aériens "plateforme" en diminuant leur coût tout en maintenant notre supériorité technologique.

## V LES CONTRAINTES FINANCIERES

Enfin, il existe une dernière contrainte et non la moindre: *la contrainte financière*. Vous n'êtes pas sans savoir que celles-ci pèsent de plus en plus lourdement sur tous les budgets militaires des pays de l'Alliance. Elle est en particulier très sensible dans le domaine qui nous intéresse. Une évaluation globale du coût de l'ensemble des senseurs et des capteurs embarqués à bord d'un vecteur piloté qui concourent, plus ou moins étroitement, à la réalisation d'un système performant de guidage de pilotage et de contrôle montre que celui-ci représente pratiquement 40% du coût total du vecteur.

Face à ces contraintes financières et à l'ampleur des études à mener à terme, il apparaît à l'évidence que l'atteinte de tels objectifs dépasse bien souvent les possibilités d'un seul pays et que le besoin de conjuguer nos efforts au sein de l'Alliance, dès que de nouvelles technologies apparaissent prometteuses, devient de plus en plus pressant.

Les négociations en cours sur la réduction des armements conventionnels et les armements stratégiques à courte portée qui conduiront à terme à la parité numérique des systèmes d'armes Est—Ouest hors des frontières du pays d'origine (USA—URSS) rendront cette nécessité de coopération encore plus nécessaire.

Il faudra non seulement coopérer pour la réalisation de tel ou tel système d'arme, mais encore reporter les efforts sur le large éventail de la panoplie des systèmes d'arme à mettre en oeuvre pour maintenir notre efficacité.

A la notion de durée face à un ennemi supérieur en nombre nous devons substituer la notion d'adaptation de nos forces à la neutralisation d'objectifs tant sur le théatre des combats qu'en profondeur ce qui conduira probablement à revoir nos armements aériens conventionnels et par voie de conséquence les concepts de guidage et de pilotage des armements des futures générations.

S'il est utopique de penser à court terme de voir disparaitre la compétition internationale industrielle dans ce secteur de pointe, il me semble réaliste, en raison de l'enjeu que représente le maintien de la liberté du monde occidental, de regrouper nos efforts de recherche et de concept afin de dégager un certain nombre de troncs communs, de mode d'action et d'interopérabilité entre les divers systèmes d'armes, tout en laissant à chacune des Nations leur Génie industriel et le libre choix de coopérer sur des réalisations aérospatiales communes.

Il est bien souvent regrettable de voir plusieurs nations de l'Alliance épuiser individuellement leurs ressources financières en recherches et réalisations sur un même problème technologique pour finir par constater que, la solution au problème posé est trouvée par chacun de ces pays, pratiquement au même moment, à de très rares exceptions près, et que le produit réalisé a sensiblement les mêmes performances et fait appel aux mêmes novations.

Dans bien des cas, sur des objectifs clairement définis et approuvés, une meilleure coopération au sein des pays de l'Alliance, dès la genèse du projet, aurait permis d'atteindre les objectifs à moindre coût et certainement plus rapidement sans compromettre l'essort industriel de nos pays respectifs.

Si dans le passé, on pouvait encore admettre que malgré les coûts élevés, tel ou tel pays de l'Alliance pouvait s'offrir ce luxe, les grands enjeux technologiques aérospatiaux des prochaines décennies (SDI — Avion de Transport Supersonique) ne le permettront plus.

Si nous poursuivions dans cette voie égoiste, il est à craindre que beaucoup de ces études resteraient des "études papier...."

Nos adversaires potentiels, plus pragmatiques, tireraient bénéfice de nos problèmes et en plus de leur supériorité numérique dès les premiers jours du conflit nous rejoindraient sur le plan technologique.

Par manque de crédits de programme ou par un étalement dans le temps de l'industrialisation des vecteurs, l'OTAN perdrait ainsi sur le plan des armes conventionnelles la crédibilité qui est encore la sienne et que je qualifierai de "Dissuasion Conventionnelle tactique et manoeuvrière" ce qui serait néfaste à la stratégie globale des pays de l'Alliance.

## VI CONCLUSIONS

L'AGARD est certainement l'organisme le mieux adapté et par le sujet qui concerne, au sein de celle-ci, votre commission pour mener à terme de telles études et convaincre vos autorités respectives de l'intérêt d'une coopération concertée.

En effet, vous êtes pratiquement le seul forum, sinon le seul groupe d'experts de haut niveau où, en dehors de toute contrainte étatique vous pouvez vous consacrer entièrement à des recherches novatrices, formuler des recommandations pertinentes et maintenir le dialogue permanent entre tous les experts des diverses disciplines scientifiques qui concourent à la réalisation des futurs systèmes de guidage et de pilotage dont vos futurs systèmes d'armes devront être dotés.

# FLIGHT CRITICAL DESIGN CONCEPTS FOR LOW-LEVEL TACTICAL GUIDANCE AND CONTROL

Michael R. Griswold
General Dynamics Fort Worth Division
Fort Worth, Texas, 76101
United States of America

## SUMMARY

Low-level combat operations (< 100 meters), such as might be typified by next generation Close Air Support (CAS) aircraft, present new demands on the guidance and control system. The design must address not only the traditional flight-critical definitions related to system management, ground collision avoidance, and operational flight restrictions, but also the possibility of increased exposure to defensive countermeasures due to system failure. It is also clear that traditional guidance and control methods must be re-examined in light of total mission goals. For instance, employing an active sensor to aid in terrain avoidance may decrease overall combat survivability due to increased detectability by threats.

This paper presents several of the elements of flight critical concepts for low-level tactical operation. This includes classical elements as well as mission-specific considerations such as threat exposure and threat evasion. In both cases, system failure may compromise safety. The mission scenario for the discussions presented in this paper is the CAS mission using a fast moving, technologically advanced aircraft.

The guidance and control strategies for the proposed application are discussed with emphasis on system integrity considerations and performance-versus-safety-issues. The previous generation of low-level guidance and control algorithms (such as the ADLAT terrain following algorithm) has been outmoded by the advent of onboard digital terrain databases. By utilizing these databases, it is possible to devise algorithms with far better performance characteristics. At the same time, reliance on the stored terrain data expands the flight critical umbrella to include the navigation system and the process for correlating the terrain database with the actual terrain. Functional partitioning must be re-examined to meet data latency requirements and minimize the distribution of the digital terrain data.

Terrain verification is a critical process for low-level operation when the digital terrain database is utilized for fundamental guidance and control information. While the use of onboard data offers many possibilities for improving the guidance and control system, it brings the need to weigh the risks of database use. The requirements for an active terrain sensor are examined. These requirements are driven by the characteristics of the terrain (and obstacles) to be measured, and by the performance and maneuvering envelope of the aircraft and the constraints on aircraft emissions.

Fault detection and management schemes are also examined. The application of previously developed system-wide integrity management design philosophies are considered for subsystem integrity monitoring and communications. These techniques are reviewed with an eye toward analytical and inductive redundancy techniques to achieve acceptable levels of detection without resorting to physical redundancy. In addition, by integrating the operation and information exchanged between several subsystems, it is possible to achieve practical fault detection strategies through estimation filters and to improve system performance during nominal operation.

## THE CLOSE AIR SUPPORT MISSION

The Close Air Support (CAS) mission has been selected as the baseline tactical environment for these discussions. The CAS mission is extremely stressful for both the pilot and the aircraft avionic systems since by nature it requires low-altitude operation in a high threat environment with demanding targeting and weapon-delivery requirements. This mission and the desirable aircraft configuration are under considerable scrutiny by the U.S. Air Force, thus making a discussion of technology applications timely.

A CAS mission is typically distinguished from other types of tactical battlefield operations by: (1) striking hostile targets that are in very close proximity to friendly ground forces, and (2) requiring close coordination and integration with these ground forces. Figure 1 depicts the main steps in a CAS mission and the principal coordinating elements. CAS aircraft operate from the Forward Edge of Battle Area (FEBA) to about 8 kilometers within enemy territory. These aircraft can play a key role in the battlefield due to their combination of speed, range, and firepower.

To achieve the level of coordination and timing required for success, a considerable part the CAS mission is dedicated to communicating mission requirements and goals and to planning the mission. This starts with the determination of initial need for support (from the Army battalion commander), tasking by the Air Support Operations Center (ASOC) and Tactical Air Control Center (TACC), enroute control, and final control and briefing

from the Forward Air Controller (FAC). Communications to and from the aircraft involve critical processes that are undergoing technological upgrades to improve reliability and reduce pilot workload.

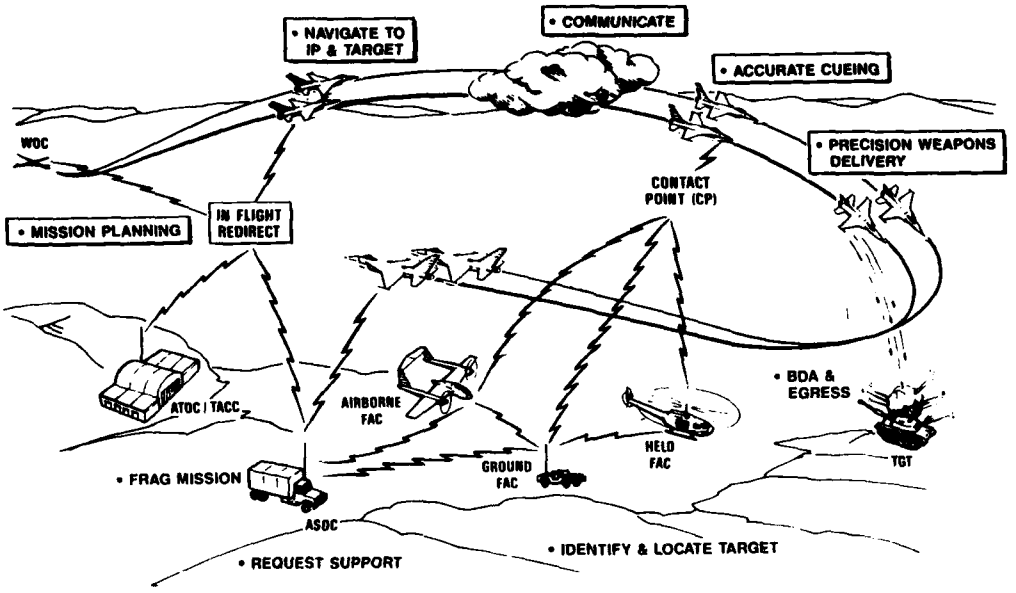



**Figure 1 CAS Mission**

*The pilot's tasks in the CAS mission*, after receiving his initial tasking, are to navigate his aircraft to the appropriate contact points, to receive his final briefing from the FAC, to acquire the target, to perform an accurate weapon delivery, and to egress. Accurate navigation is essential to reach the contact points accurately and within the given time windows. It is also a prime factor in the pilot's ability to locate the target once he has received location cues from the FAC. Weapon delivery must be precise in order to be effective against armored or hardened targets and to avoid damage to friendly forces. Situation awareness and low-altitude flight is required in order to minimize the risk from enemy threats. Operation at night further compounds the pilot's workload.

While the configuration of the next generation CAS aircraft is still a subject of considerable debate, one of the primary candidates is an F-16 derivative. Proponents of the F-16 foresee the pilot relying on high-speed, low-altitude ingress to avoid enemy threats. To be able to ingress fast and low and still to acquire the targets, the CAS F-16 would use a specially tailored sensor suite, coupled with an accurate navigation. Finally, CAS-specific weapons would complete the vehicle configuration.

The AFTI/F-16 CAS Program is currently investigating technology applications for improving autonomous navigation and target acquisition for the F-16 or similar aircraft. (Reference 1) To date, little has been done in demonstrating automated guidance and control applications for fast moving, low-flying CAS aircraft. The upcoming AFTI/F-16 CAS flight test demonstrations may address these issues. The concepts presented here would be applicable to this effort.

## GUIDANCE AND CONTROL ISSUES

In developing a guidance and control strategy for an F-16 class CAS aircraft, several major issues must be considered. Perhaps the first question concerns the proper level of automation that should be provided. Automation is typically used in a fighter for two purposes: (1) to help alleviate pilot workload during critical mission phases, and (2) to perform tasks which might be outside the pilot's control bandwidth. Of course these two aspects are related to some extent since high workload may reduce the pilot's effectiveness in performing some "high-gain" tasks. For the CAS mission, clearly the pilot's work-

load is high. Communications, coordination with other friendlies, target acquisition, and threat avoidance will quickly work to saturate the pilot's attention. If low-level, high-speed tactics are employed, the pilot's control bandwidth may also become saturated if significant terrain is encountered. This may cause the pilot to fly higher, raising the level of threat exposure. Given these considerations, some level of guidance and control automation is reasonable. If we consider a fully automated guidance and control system for CAS, several of its fundamental characteristics can be readily deduced.

Since CAS operations, by their very nature, involve delivering weapons on enemy targets that are in close proximity to friendly troops, the weapon-delivery system must be precise. Weapon lethality is a function of target type, weapon effectiveness and delivery accuracy. Since target types are specified, improved weapons and delivery accuracy are the primary considerations for improvements. The key for survivability is to minimize the number of passes that must be made by making every pass count.

Both passive and active threat avoidance are also important in defining the guidance and control strategy. Passive avoidance is achieved by maintaining the least exposure to potential threat's (known or otherwise) by flying low to reduce the threats effective horizon, and by flying as fast as possible to deny the threat enough time to react to ownship detection if it occurs (Figure 2). To achieve these goals implies a terrain following capability that not only has good terrain hugging characteristics but does not of itself limit the aircraft penetration speed. Despite the advantages of low flight, there remains a practical limit on how low the aircraft may reasonably operate (Figure 3). The nature of the ground collision curve is very much a function of the guidance and control scheme employed.

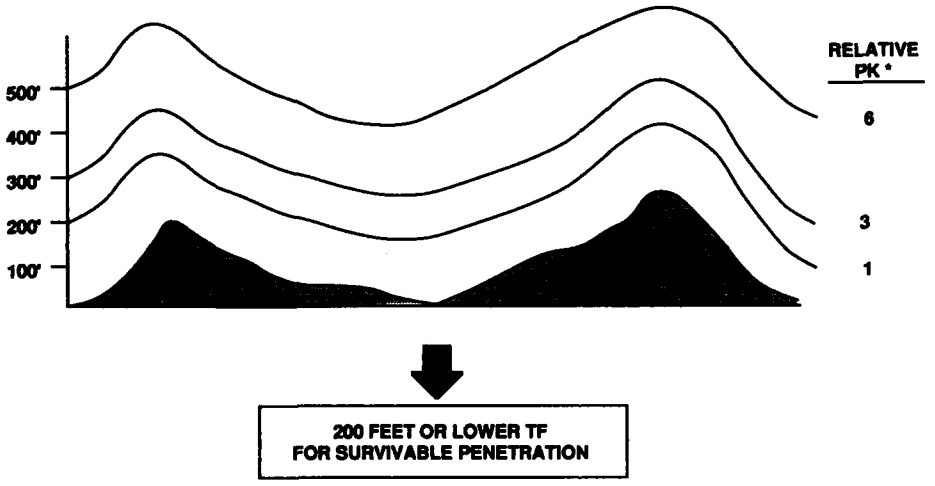



Figure 2 Aircraft Survivability Is Altitude Dependent

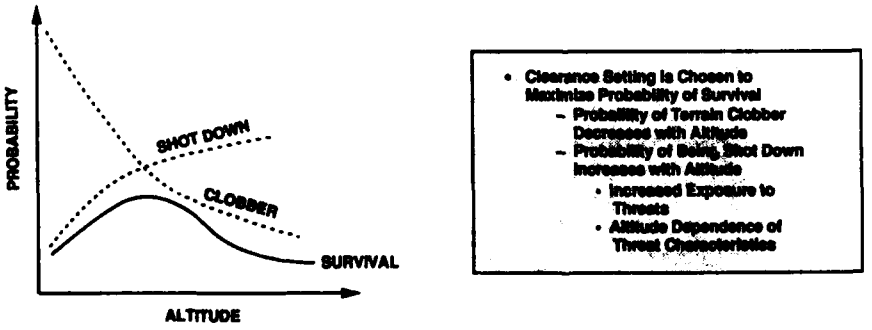



*Figure 3 Terrain Clearance Constraints*

Active threat avoidance implies actually determining a route through an array of known or suspected threats, and accounting for threat sensor characteristics and lethality. This typically involves using modeled threat data, terrain profiles, course constraints (checkpoints, contact points, etc.), target location, and possibly schedule constraints (time over target, free fire zone windows). While systems have been demonstrated with these capabilities, ground-based in particular, systems suitable for tactical fighters are only now beginning to emerge. The practical constraint on these systems is to provide sufficient processing power onboard to allow the near-real-time replanning required to effectively handle unexpected threats. In particular, if the route planning system is part of an automated guidance and control system, it clearly must not be allowed to steer the aircraft into known threats, either by virtue of slow response or system failure. A complement to the route planning guidance scheme for threat avoidance is an automated evasive maneuver taken in response to a threat detection (or even launch indication).

In summary the guidance and control system for this conceptual CAS aircraft provides automated systems for terrain following, optimal route planning for threat avoidance, and weapon delivery. The terrain following algorithm should provide robust terrain hugging capability to generally limit threat exposure.

The system should not limit the pilot's ability to maneuver aggressively while still assuring ground clearance. The route planning algorithms must be responsive and accurate in minimizing threat exposure. Weapon delivery, the ultimate goal of the mission, must be accurate to achieve the desired damage to the enemy while avoiding friendly forces. These functions are flight critical since failure or poor performance can cause the aircraft to hit the ground, be shot down, or drop weapons on friendly forces.

Recent General Dynamics' experience with low-level automated guidance and control was successful in developing an automated system for ground attack, which was demonstrated on the AFTI/F-16 Program (Reference 2). While this system clearly showed the potential for this type of operation, the system was constrained to operation over relatively flat terrain (less than 2% grade). This allowed the development of a reliable ground collision avoidance system with a straight-forward combination of radar altimeter and inertial measurements. Clearly, this is insufficient for CAS operations, which will likely be required in rugged terrain.

**TERRAIN DATA MANAGEMENT**

The key to extending the previous AFTI/F-16 development is the application of onboard digital terrain data. The database provides an independent source of elevation data that can be used and correlated with other (real-time) measurements to give an unprecedented picture of the surrounding terrain. In general, the database can be considered a sensor; its limitations, such as accuracy and failure modes, must be fully accounted for. The database also provides a convenient logical representation of terrain data for the correlation and blending of data from other sources.

The decision to rely on the digital terrain database as a primary source of terrain information (as opposed to a radar) is not without problems (see Figure 4). The principal issues cited are the accuracy and completeness of the database. While these problems should be reduced with time (due to improved mapping techniques), their impacts must be carefully weighed against the probability of loss of the aircraft, particularly during peacetime.

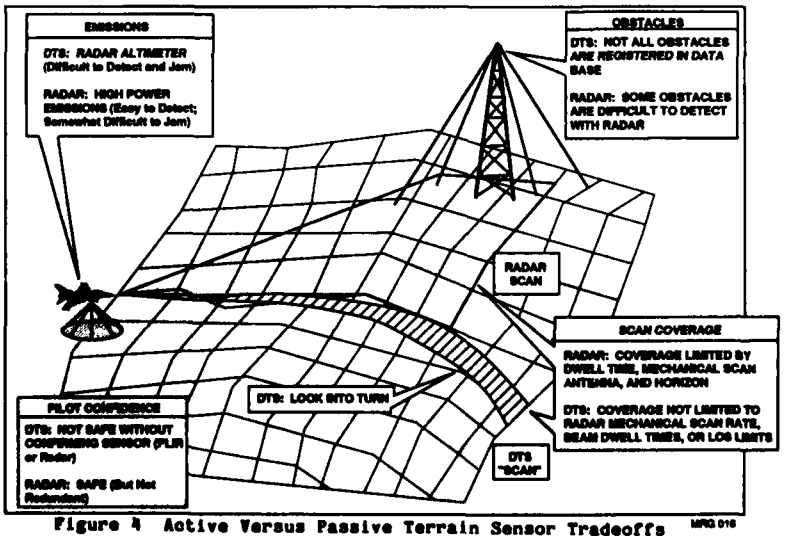



**Figure 4 Active Versus Passive Terrain Sensor Tradeoffs**

Terrain data management encompasses three related processes, terrain data correlation, terrain data verification, and terrain data blending. Reliance on a stored terrain database requires the proper correlation of the actual aircraft position within the database. This registration can be accomplished by either a self-contained terrain correlation algorithm, by an external position fixing system such as the Global Positioning System (GPS), or a combination of both. Reference 3 describes several representative algorithmic approaches. While the original motivation for these algorithms was increased navigational accuracy, they now form the basis for referencing the actual aircraft position with the stored terrain database. Since data are being extracted from the database for guidance and control purposes (i.e., terrain following), the correlation process is clearly flight-critical. Unfortunately, the current implementations of these algorithms are not fail-safe. Architectural considerations aside, the correlation algorithm's own estimate of its performance is not an adequate failure monitor.

Even a highly registered terrain database may not accurately represent the surrounding terrain. Several General Dynamics' sponsored flight test experiments have pointed up various classes of database anomalies such as rounded or truncated peaks, shifted terrain features, and actual missing features. In addition, significant man-made obstructions (towers, power lines, etc.) are not reliably represented. Even if the pilot is manually flying the aircraft, it cannot be assumed that he will perceive a potential collision in time to react. Some form of real-time data-base verification is required to overcome these problems. (Of necessity, a course of action must be planned in the event discrepancies are noted.)

The probability of collision for a low-flying aircraft is dependent on: (1) the probability of actually encountering an obstacle, (2) the probability of not detecting the obstacle once it is encountered, and (3) the probability of colliding with the obstacle if it is not detected (or not detected in a timely manner). Critical obstacles can typically be categorized as terrain, towers (and possibly its related support wires), and cables spanning towers. Various studies have attempted to quantify the encounter rate for different types of obstacles (Reference 4).

Establishing the obstacle encounter rate for various types of obstacles is a critical design point since it strongly influences the type of sensor required for obstacle detection. In particular, the percentage of obstacles that Ku- and X-band radars cannot reliably detect becomes significant within 100 meters of the ground. The probability of obstacle detection is a function of range to the obstacle. Minimum acceptable range can be determined from the sum of the maneuver time (for obstacle avoidance) and the measurement and processing times. These detection ranges typically vary from 1 to 3 kilometers depending on the obstacle type. Obviously the use of an active sensor to reduce terrain database risk itself increases the risk of detection by potential threats (Figure 5). For this reason, techniques are evolving for so-called covert sensors which rely on low-power levels, modified wave forms and different operating frequencies to avoid detection. Table I gives typical sensor parameters.

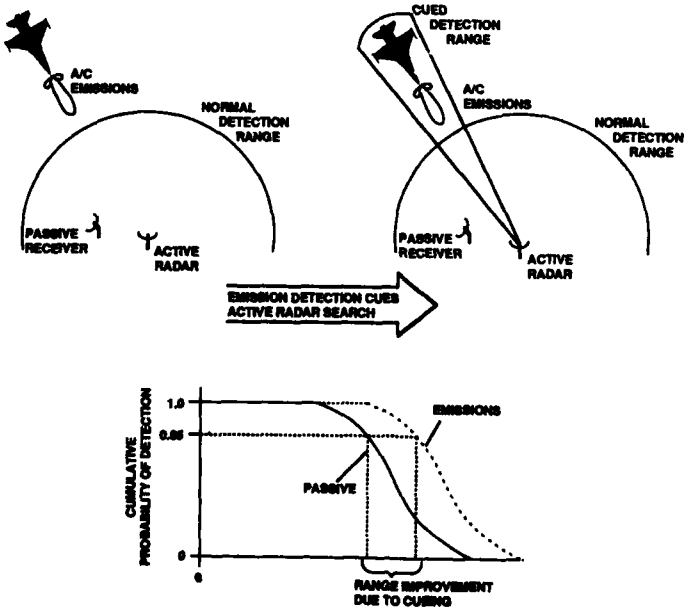



**Figure 5 Active Sensor Emissions Decrease Survivability**

**Table I Guidance and Control Functions**

| |
|---|
| • DRIVEN BY OBSTACLE DETECTION RANGE, COVERTNESS AND SCAN VOLUME |
| • POWER AND SCAN MANAGEMENT LIMIT DETECTION<br>– TYPICAL FIGHTER ANTENNA - 3° BEAM WIDTH<br>– 25dB PROCESSING GAIN FROM SPREAD SPECTRUM CODING<br>– OXYGEN ABSORPTION LINE - 1-2 WATTS<br>– MM WAVE 5-6 WATTS<br>– $CO_2$ LASER - 7.5 cm APERTURE<br>- 10 WATTS AVERAGE CLEAR WEATHER<br>- 100 WATTS AVERAGE FOG (3dB/Km FOG 0.5 MILE VISIBILITY) |
| • SCAN VOLUME – 25° VERTICAL<br>– 25° HORIZONTAL FOR 5.5°/SEC TURN<br>– 60° HORIZONTAL FOR 12°/SEC TURN |

To provide a more robust system, the stored terrain data and the sensed terrain data should be combined to provide a composite database from which to drive the various guidance and control algorithms. In the process of combining the terrain data, differences in the measured elevation from the different sources can be compared against established thresholds to determine if an exceptional condition (such as sensor failure) has occurred. The simplest approach is to use the highest of either the sensed or the stored terrain elevation at a given point. This is the most conservative approach, but it runs the risk of driving the aircraft higher than needed for terrain clearance and may increase threat exposure. A maximum-likelihood blending algorithm would reduce the conservatism by combining the elevation data on the basis of their respective measurement variance.

## GUIDANCE AND CONTROL ALGORITHMS

A conceptual representation of the guidance loop is shown in Figure 6. In this model, the terrain data management and the guidance command generation functions represent new flight-critical elements. By using the terrain database format as the underlying representation of terrain data, the guidance and control algorithm development can be largely decoupled from the sensor development. (This is not to say that the choice of sensors is unimportant to the algorithm designer, indeed the available sensor suite may drive the fundamental guidance and control strategy.) Typically, three basic guidance and control algorithms are needed. The first is a terrain-following algorithm that provides vertical terrain clearance. The second is a threat and/or obstacle-avoidance algorithm that provides lateral steering. The third is a ground-collision-avoidance algorithm that provides a preemptive response to impending ground impact. Other candidate algorithms might include an automated weapon-delivery system and a missile evasion system. The performance criteria of these algorithms must be established in light of the CAS mission. Criteria that apply for strategic missions or interdiction missions may not be appropriate.

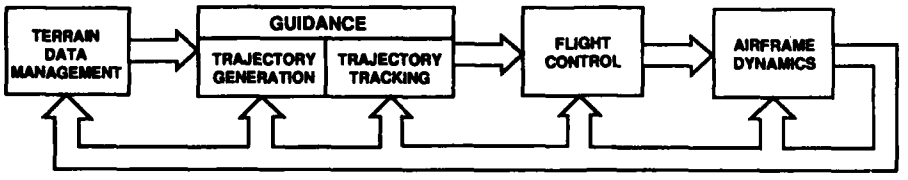



**Figure 6 Terrain-Based Guidance Loop**

For CAS, the terrain following system must be dynamic enough to allow extensive pilot maneuvering. Currently fielded terrain-following implementations typically limit the aircraft turn rate on the order of 5 degrees per second. This constraint, while largely driven by the terrain sensor characteristics, is not incompatible with their design mission. Since emerging agile beam radars and scanning laser ranging systems, coupled with the digital terrain database, can remove these limitations, the algorithmic design can be readdressed. The particular need is to provide a wide maneuvering envelope consistent with the CAS mission requirements.

The need for an independent ground collision avoidance system (even in the presence of a terrain-following system) is two-fold. First, an independent algorithm can provide coverage for certain classes of failures in the terrain-following system (depending on the overall system architecture). More importantly, if the pilot is manually flying the aircraft through a maneuver which is outside the normal terrain-following envelope (such as missile evasion), the ground collision avoidance system should operate to provide a preemptive safeguard against ground collision. In essence, the two algorithms, terrain-following and ground-collision-avoidance, have subtly different goals. The terrain- following algorithm attempts to maintain a set clearance along a certain flight path. The ground-collision-avoidance algorithm attempts to prevent ground collision for any state the aircraft attains.

## INTEGRITY MANAGEMENT

For integrity management purposes, the architecture of the guidance and control system is shown in Figure 7. Typically, single-thread sensors and systems are supplying data to a system manager. The manager, while single thread computationally, has access to enough multiple-sensor measurements to judge the validity of its data sources. Finally, a physically redundant system judges the health of the entire system. Previous General Dynamics' experience in the design of fault detection algorithms for the AFTI/F-16 AMAS low-level operations (Reference 5) was successful in developing a robust and reliable system by using this architecture.

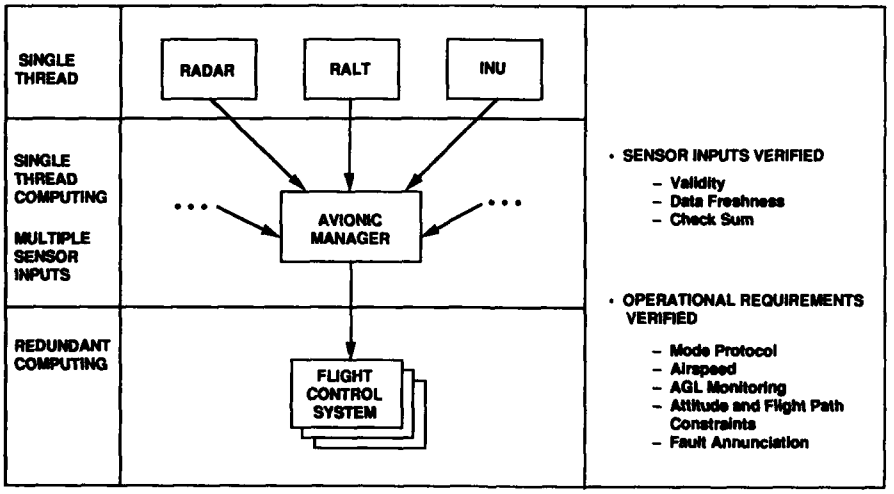



**Figure 7 Multilevel Integrity Management Architecture**

Even a poorly conceived system may be reliable during normal operation. A thoroughly reliable system could be designed using a stored terrain database, terrain-following system driving the flight control system for automated operation. Without a proper understanding of the potential failure modes and, just as importantly, of their effects, the system cannot be considered safe.

High-speed maneuvering in rugged terrain or in close proximity to the ground will demand a rapid assessment of systemic hazards resulting from malfunction or miscalculation. Potential hazards can result from (1) failure or inappropriate operation of physical elements (such as hydraulics, processors, media used for storage and communication, and sensors), and (2) algorithmic and implementation flaws, as well as inadvertent pilot actions. After detecting a hazardous situation, the integrity management system must provide for the safe recovery from the situation and for an orderly resumption of manual pilot control. In order to provide for the detection of suspected subsystems and for the proper identification and annunciation of faults, the integrity management system must provide at least a single-fail-safe capability. In general, the system must rely upon various redundancy techniques.

Various redundancy techniques can be applied to the design to meet the single-fail-safe criteria. The first and most obvious technique is physical redundancy. This is used in modern digital flight control systems to provide fail-operate capability. Because of this redundancy, the flight control system is a logical choice as the overall integrity manager of the guidance and control system. Physical redundancy is not practical for the majority of the avionic suite however. Other techniques are functional redundancy - identical processing in different hardware, temporal redundancy - the same processing done at different times, and inductive redundancy, using dissimilar sensors or processes.

The proliferation of complementary data sources and estimation processes onboard the aircraft opens the door to a variety of error estimators (filters) that should improve the safety and robustness of the guidance and control system. By monitoring Kalman processing residuals and input measurements, fault detection and identification of the various data sources is possible.

Establishing the criteria for failure declaration must consider the tradeoff between false alarms and catastrophic failures. The acceptable loss rate for tactical aircraft is typically specified by the operating service. For strategic terrain-following systems, a false alarm rate of 1 per hour is standard. For the CAS mission, the false alarm rate has not been established. Only when these two end points have been established can the fault detection scheme be completely specified.

## ACKNOWLEDGEMENTS

The author would like to acknowledge D. W. Schaefer for his paradigm of a terrain-relative guidance and control system and T. P. Kelley for sensor parameter data.

## REFERENCES

1. Baird, C. A., Collins, N. and Drew, M., Terrain-Aided Navigation and Target Acquisition on the AFTI/F-16, NATO/AGARD Guidance and Control Panel, 48th Symposium on Advances in Techniques and Technologies for Air Vehicle Navigation and Guidance, Lisbon, Portugal, May 1989.

2. Griswold, M. R., AFTI/F-16 Automated Maneuvering Attack System Guidance and Control, NAECON-86, Dayton, OH, May 1986.

3. Baird, C. A., Abramson, M. R., A Comparison of Several Map-aided Navigation Techniques, IEEE Position Location and Navigation Symposium, San Diego, California, November 1984.

4. Sensor Blending For Terrain and Obstacle Avoidance, AFWAL-TR-84-1173, Vol I, March 1985.

5. Gordon, F. W., Johnston, A. M., Barfield, A. F., AFTI/F-16 Automated Maneuvering Attack System Safety Design, NAECON-86, Dayton, OH, May 1986.

# Evolution dans les applications civiles

Pascal Traverse

Aérospatiale
316, route de Bayonne
F-31060 Toulouse CEDEX 3

**Résumé**

Les commandes de vol de l'Airbus A320 marquent une étape dans l'histoire de l'aéronautique, comme dans l'histoire des systèmes informatiques tolérants aux fautes. Ce texte présente ce système, ainsi que des évolutions possibles de ce type de système. Les évolutions plus particulièrement détaillées concernent l'architecture des calculateurs, l'architecture informatique du système, l'utilisation de l'optique, et les méthodes de conception de systèmes.

## Introduction

Le premier système de commandes de vol électriques sur avion civil a été conçu par l'Aérospatiale et installé sur Concorde. Ce système est analogique, a pleine autorité sur toutes les gouvernes et effectue une recopie des ordres manche sur les gouvernes. Un secours mécanique existe sur les trois axes.

La première génération de systèmes de commandes de vol électriques, et de technologie numérique est apparue sur plusieurs avions civils au début des années 1980, sur l'Airbus A310, entre autres. Ces systèmes controllent les becs, les volets, et les spoilers. Ces systèmes ont des exigences de sécurité sévères (l'embarquement de ces surfaces doit être Extrêmement Improbable). Par contre, la perte de fonction est admise, car n'ayant pour conséquence qu'un accroissement supportable de la charge de travail de l'équipage.

L'Airbus A320 est le premier exemplaire d'une deuxième génération d'avions civils à commandes de vol électriques. Sa particularité est que toutes les surfaces sont controllées électriquement avec des lois de pilotage évoluées en fonctionnement normal, et que le système a été conçu pour être disponible en toute circonstance. Les projets connus de commandes de vol électriques pour avion civil n'apportent pas d'avancée significative par rapport à l'A320.

A moyen terme, des changements au niveau des calculateurs sont prévisibles, ainsi que des évolutions de l'architecture informatique des commandes de vol. En particulier, les commandes de vol peuvent devenir un système informatique distribué. La nécessité d'un secours de technologie non digitale est discutable. Néanmoins, cette nécessité est prise en compte par l'Aérospatiale, et l'utilisation de l'optique à cet effet est envisagée. Les méthodes de conception de systèmes sont également en phase d'évolution. Les efforts de l'Aérospatiale dans le domaine sont présentés.

## 1. Les commandes de vol électriques de l'A320

Les commandes de vol de l'A320 ont été décrites par ailleurs (ref. 1, 2 , 3 , 4). Nous ne les traiterons que d'un point de vue sûreté de fonctionnement. Dans le principe (figure 1), le système des commandes de vol est composé d'organes de commande (manches latéraux, levier d'aérofrein, ...), de calculateurs, de capteurs de la position de l'avion (centrales à inertie et barométriques, accéléromètres), et d'actionneurs. Les calculateurs asservissent les actionneurs. La consigne d'asservissement est une fonction de la position du manche (et donc de la demande exprimée par le pilote), et des retours avion.

Il est possible de distinguer trois grands groupes de fonctions : 1) interface avec l'équipage (acquisition et surveillance des organes de pilotage, information sur la position des surfaces et l'état du système), 2) fonctions liées au lois de pilotage (gestion des information inertielles et barométriques, calcul des lois, en particulier pilotage en facteur de charge, amortissement du roulis hollandais, coordination de

virage, protection haute incidence, ...), 3) contrôle des gouvernes de l'avion sur les trois axes (roulis, lacet, tangage), et aérofreinage.

Un des apports des commandes de vol électriques à la sécurité de l'avion tient aux protections qui sont partie intégrante des lois de pilotage. Ainsi, en pilotage normal, la structure est protégée (facteur de charge, vitesse). Une troisième protection, dite haute incidence, évite à l'avion de décrocher. Ces protections déchargent le pilote, en particulier lors de manoeuvres d'évitement, que ce soit d'un obstacle (quasi-collision avec un avion, "near-miss"), ou d'un cisaillement de vent ("windshear"). Ces protections apportent une sécurité accrue. Ainsi, un pilote qui doit éviter un autre avion peut se concentrer sur la trajectoire à suivre, sans se soucier des limites structurales de l'avion, ou d'un éventuel décrochage. Un cisaillement de vent se produit généralement à faible altitude. La réaction sûre est délicate à effectuer dans la mesure où il ne faut surtout pas que l'avion décroche. Le fait que l'incidence de l'avion est automatiquement contrôlée, couplé à un accroissement automatique du régime moteur à grande incidence, apporte à l'A320 un accroissement significatif de probabilité de survie à un cisaillement de vent. Pour apprécier pleinement l'intérêt d'une telle protection, il n'est que de rappeller que sur les 5 dernières années, 2/3 des personnes tuées dans un accident d'avion aux Etats-Unis l'ont été à la suite d'un cisaillement de vent (voir ref. 5).

Un premier type de défaillance à prendre en compte est une défaillance matérielle des équipements du système. Les calculateurs sont à commande et surveillance, ce qui permet de rendre Extrêmement Improbable un embarquement de gouverne par un calculateur.

### 1.1. Architecture des calculateurs

Les calculateurs utilisés pour les commandes de vol de l'A320 sont à commande et surveillance. Ce type de calculateur est largement utilisé sur les avions Airbus A300, A310, tant pour des fonctions de commandes de vol que de commande automatique du vol. La partie commande assure la fonction attribuée au calculateur (contrôler des gouvernes en particulier). La partie surveillance sert à assurer un fonctionnement correct de la partie commande. La comparaison des résultats est effectuée dans les deux chaînes. Cette comparaison est réalisée en logiciel. Ces calculateurs sont construits autour de deux chaînes de calcul (figure 2), qui chacune compare ses résultats avec ceux de l'autre (figure 3). Chaque chaîne comprend un ou plusieurs processeurs, leur mémoire associée, des circuits d'entrée/sortie, et un bloc d'alimentation. Quand les résultats d'une de ces deux chaînes diverge sensiblement, la (ou les chaînes) qui a détectée cette erreur interrompt les liaisons entre le calculateur et l'extérieur. Le système est ainsi fait que les sorties du calculateur sont alors dans un état sûr. La détection d'erreur se fait essentiellement en comparant l'écart entre les ordres de commande et de surveillance avec un seuil pré-établi. Ce schéma permet donc de détecter les conséquences d'une défaillance d'un des composants du calculateur, et d'empêcher à l'erreur résultante de se propager hors du calculateur. En pratique, l'actionneur, du point de vue de cette surveillance, est inclus dans la chaine de commande. Ce moyen de détection est généralement complété par une surveillance de la bonne exécution du programme, au travers de son séquencement (enchaînement des tâches, et durée). Cette surveillance se fait par des échanges d'information entre processeurs (dans le cas d'une chaîne de "commande" bi-processeur), ou encore grâce à un "super" chien de garde (ce chien de garde est ainsi qualifié pour marquer sa différence par rapport aux chiens de garde les plus utilisés qui ne surveillent que la capacité du processeur à émettre à intervalle fixe un signal donné). De plus, des tests de vraisemblance sont effectués pour vérifier la validité de certaines données.

Ce schéma pourrait être mis en défaut par une erreur produite à la fois dans la partie commande et dans la partie surveillance. Un premier point commun pourrait être constitué par le logiciel. En effet, si ce logiciel est le même et qu'il contienne des fautes, on peut s'attendre à ce que ces fautes produisent des erreurs tant en commande qu'en surveillance, ce qui n'est pas nécessairement détecté par une comparaison des résultats. La méthode de base pour traiter ce problème est d'écrire les logiciels avec un soin particulier. Règlementairement (et donc avec une grande sévérité), ces logiciels répondent aux normes les plus exigeantes de l'aviation civile (logiciel niveau 1 - ref. 6), et ceci est suffisant. De plus, ils subissent une somme considérable d'essais.

Une précaution supplémentaire est d'utiliser en commande un logiciel différent de celui utilisé en surveillance. La terminologie utilisée en l'occurrence est riche, mais le terme que nous

employons est "dissimilarité". Le but est d'éviter qu'une même faute soit présente dans ces deux logiciels. Le principe utilisé pour avoir deux logiciels dissimilaires est d'avoir deux chaînes de production de logiciel différentes. Ainsi, une défaillance d'un des éléments d'une chaîne (d'un programmeur, par exemple) ne doit pas avoir de conséquences dans les deux logiciels. De plus, des règles de programmation sont utilisées qui visent à accroître la dissimilarité, en particulier quand un point paraît complexe (voir ref. 7).

L'environnement du calculateur pourrait être un autre point commun. Les chaînes de commande et surveillance du calculateur ont la même source électrique, le réseau électrique de l'avion (28VDC). Elle doit être convertie et régulée à l'intérieur de chaque chaîne de commande et de surveillance. Ces alimentations sont doublées, chacun des blocs étant associé à une chaîne de calcul. Ainsi, les modes communs de défaillance du système d'alimentation sont détectables. Le cas le plus probable est la perte d'alimentation du calculateur. La conception retenue place ce dernier dans un état sûr. Le calculateur est également protégé contre les possibles sur/sous tensions, ainsi que contre les perturbations électro-magnétiques et les effets indirects de la foudre. Ces protections couvrent toutes les agressions que l'avion est susceptible de rencontrer. Cette protection est assurée par un filtrage de tous les fils sensibles entrant ou sortant du calculateur. De plus, les cables sont également protégés (blindage, torsade).

Une protection supplémentaire consiste à ne pas synchroniser strictement les chaînes de commande et de surveillance, d'introduire une séparation physique entre les deux chaînes, et de concevoir le système de telle manière que les chaînes aient des entrées différentes. L'objectif est que si le calculateur est perturbé malgré ses protections, alors la commande et la surveillance sont affectées dans des états différents, et ainsi leurs sorties sont différentes et la perturbation est alors détectée et passivée.

Certaines défaillances peuvent rester masquées longtemps après leur création. Ceci est typiquement le cas de la passivation d'une chaîne de surveillance qui n'est détectée que lors de la défaillance de la chaîne surveillée. Des tests sont pratiqués périodiquement pour que la probabilité d'occurrence d'un événement indésirable reste suffisamment faible. Typiquement, un calculateur s'auto-teste et teste ses périphériques lors de la mise sous tension de l'avion, donc au moins une fois par jour. Le but est d'être exhaustif pour les pannes les plus dangereuses. Des tests en-ligne sont également effectués (par exemple, un calculateur peut pratiquer un check-sum de sa mémoire morte en permanence).

### 1.2. Architecture du système

Une défaillance de calculateur va donc se traduire par un arrêt de celui-ci. Les actionneurs sont surveillés par les calculateurs, tant par la chaîne de surveillance du calculateur, que par la chaîne de commande. L'une et l'autre chaîne peuvent passiver l'actionneur. Une autre source d'embarquement est constituée par les différents capteurs (sur les manches, les actionneurs, les centrales à inertie, ...). Chaque capteur est au moins dupliqué, de manière à ce que toute information utilisée soit consolidée par comparaison entre au moins deux sources d'information diférentes.

Le système étant protégé contre les embarquements, doit donc être construit pour être suffisamment disponible et donc suffisamment redondant. L'électricité est normalement fournie par deux alternateurs, chacun étant entraîné par un moteur différent (figure 4). En outre, des batteries et un générateur auxiliaire (APU) sont disponibles, ainsi qu'une éolienne. En cas d'arrêt des deux moteurs, cette éolienne se déploie automatiquement. Elle pressurise alors un circuit hydraulique, qui entraîne un troisième générateur électrique. Les calculateurs ne sont pas liés à une seule source d'énergie, mais à au moins deux. L'avion compte trois circuits hydrauliques, quant un seul suffit pour contrôler l'avion. Deux circuits sont pressurisés par un moteur chacun, le troisième l'étant par une pompe électrique, ou encore par l'éolienne. Les calculateurs et actionneurs sont également redondants. Ceci est illustré par le contrôle en tangage de l'A320 (figure 5). Quatre calculateurs à commande et surveillance sont utilisés (ELAC : ELevator and Aileron Computer, SEC : Spoiler and Elevator Computer), un seul suffit à contrôler l'avion. En fonctionnement normal, un des calculateurs (ELAC2) contrôle la profondeur. Les autres calculateurs contrôlent d'autres surfaces. Si l'ELAC2 ou un des actionneurs qu'il commande tombe en panne, l'ELAC1 prend le relais. Suivant le même mode de défaillance, l'ELAC1 peut avoir à passer la main

au SEC2. De même le contrôle de la profondeur passe d'un SEC à l'autre, en fonction du nombre de surfaces qu'un de ces calculateurs peut commander. Il est à noter que 3 calculateurs seraient suffisants pour tenir les objectifs de sécurité. Le calculateur supplémentaire est pleinement justifié par des contraintes opérationnelles : il est souhaitable de pouvoir tolérer une impasse technique sur un calculateur (*décoller avec un calculateur en panne*).

Le système des commandes de vol a suivi un processus de conception et de fabrication très exigeant et dont on peut raisonnablement estimer qu'il assure un niveau de sécurité largement suffisant. Une protection supplémentaire a néanmoins été prise, qui consiste à utiliser deux types de calculateurs différents : les ELAC réalisés par Thomson-CSF, autour de microprocesseurs Motorola, et les SEC dont le matériel est à base de microprocesseurs Intel et construit en coopération SFENA/Aérospatiale. Nous avons donc deux équipes différentes de conception et de fabrication, avec des microprocesseurs (et circuits associés) différents.

L'installation électrique, en particulier les multiples liaisons électriques, posent également un risque de points communs. Ceci est évité par une ségrégation poussée : en fonctionnement normal, deux systèmes de génération électrique existent et n'ont aucun point commun. De plus, les liaisons qui servent à la surveillance ne cheminent pas avec celles utilisés par la commande. La destruction d'une partie de l'avion est également prise en compte : les calculateurs sont répartis en trois endroits différents, certaines liaisons vers les actionneurs passent dans le plancher, d'autres au plafond, et les dernières en soute.

Malgré toutes ces précautions un secours mécanique a été conservé sur le Plan Horizontal Réglable et la gouverne de direction. Via le Plan Horizontal Réglable (figure 5) il permet de contrôler l'axe de tangage. Via la gouverne de direction, il permet de contrôler directement l'axe de lacet, et indirectement l'axe de roulis.

## 2. Evolution des commandes de vol

L'architecture informatique des commandes de vol des avions civils en développement ou en projet diffère peu de l'A320, d'un point de vue sûreté de fonctionnement. Les Airbus A330/A340 (premier vol en 1991) ont un système des commandes de vol qui est une adaptation à ces avions du système A320, avec en particulier les mêmes principes quant à l'aspect sûreté et tolérance aux fautes. Deux avions soviétiques ont des commandes de vol électriques (ref. 8, le Tupolev Tu-204, premier vol en janvier 1989, et ref. 9, l'Ilyushin Il-96-300, premier vol en octobre 1988), le projet Boeing B7J7 (ref. 10) également. Plutôt que des calculateurs à commande et surveillance, ces avions utilisent des calculateurs triplex, le Tupolev et le Boeing ayant la particularité de ne pas avoir de secours mécanique, mais un secours à base de chaines de commande analogiques.

A plus long terme, l'évolution de l'architecture informatique des commandes de vol peut être liée à l'évolution des fonctions du système d'une part, et à l'évolution de la technologie d'autre part. En parallèle, les méthodes de conception et de validation de cette conception évoluent également. L'aéronautique est un secteur d'activité en perpétuelle évolution, cette évolution se faisant de façon incrémentale et devant être profondément validée avant mise en service. Ceci sera illustrée par l'introduction de manches latéraux sur l'A320 (voir § 2.4).

### 2.1. Evolution des fonctions du système des commandes de vol

Deux types de fonctions supplémentaires peuvent être envisagées, selon qu'elles ont pour objectif de diminuer les charges structurales, ou bien sont plutôt liées à la recherche de qualités de vol différentes. La diminution des charges structurales peut être faite via des fonctions du type de la fonction d'atténuation des charges en rafale de l'A320, ou de la fonction d'atténuation des charges en manoeuvre de l'A340. Ces fonctions ne devraient pas remettrent fondamentalement en cause les principes utilisés pour concevoir les systèmes de commandes de vol actuel. Une fonction nouvelle pourrait avoir pour objectif d'amortir du flottement ou certains modes structuraux ("fish tailing" par exemple). Un problème potentiel est le réglage et le principe de surveillance d'une telle fonction. En effet, ce type de fonction commande de faibles mouvements de la gouverne, qu'il peut être difficile de

distinguer de possibles bruits ou imprécisions de capteurs.

Il peut également être envisagé d'avoir des avions naturellement instable. Ceci pourrait remettre en cause l'existence d'un secours mécanique, sans amortissement artificiel. Diverses solutions sont envisageables. Dans un premier temps, il est nécessaire de décider si un sous-système de secours est nécessaire ou non. Ensuite, la technologie de ce secours est à choisir.

Plus globalement, une certaine tendance de nos clients (les compagnies aériennes) est de suggérer que les commandes de vol réalisent des fonctions qui s'apparentent à celles du système de commande automatique du vol. L'objectif est de rendre ces fonctions plus disponibles. Ceci pourrait aboutir à une réorganisation des systèmes de commandes de vol, et de commande automatique du vol.

### 2.2. Evolution des technologies

Un autre facteur d'évolution est l'apparition d'innovation de la technique, en particulier de l'informatique, et qui soit adaptée ou adaptable à l'aéronautique. Une première évolution est liée aux sources de puissance. L'apparition d'actionneurs à puissance électrique peut conduire à la suppression d'au moins un circuit hydraulique. En contrepartie, ces actionneurs auront un impact sur le système de génération et distribution électrique. La création d'un réseau électrique spécifique aux actionneurs peut être liée à l'apparition de ceux-ci. Quant à l'informatique embarqué, il est possible de citer, en particulier l'évolution du génie logiciel, l'intégration de plus en plus poussée de fonctions sur un unique circuit intégré, l'apparition de modules avioniques standards (ref. 11), et de bus numériques à accès multiplexé (ref. 12). Les premières tendances influent plutôt sur l'architecture des calculateurs, la dernière sur l'architecture du système. L'Aérospatiale a donc lancé un ensemble d'études technologiques dont certains sont susceptibles d'influer sur les commandes de vol :

- structure de l'électronique embarquée (programme IDEE)
- système de génération et distribution électrique (programme EGIDE)
- communication à base de fibres optiques (programme ELOISE)
- actionneurs (programme CDVF)

Ce dernier programme de recherche comporte également un volet d'études orientées système. Ces études ont en particulier pour objet de prendre en compte d'éventuelles évolutions des fonctions du système, ainsi que les évolutions mises à disposition par les études à caractère technologiques, ou encore d'autres études comme celles traitant de l'interface homme-machine (programme EPOPEE/PREFACE).

#### 2.2.1. Evolution des calculateurs

L'accroissement de capacité fonctionnelle des circuits intégrés peut avoir deux conséquences. Tout d'abord, il peut devenir économiquement intéressant de délocaliser une partie des traitements, par exemple au niveau d'un capteur (sonde d'incidence par exemple), ou d'un actionneur. Cette évolution, couplée à l'apparition de bus numériques à accès multiplexé peut conduire à décharger les calculateurs centraux et à les banaliser, ce qui va dans le sens d'une utilisation dans les commandes de vol de modules (unité centrale, mémoire, entrées / sorties) standards - répondant à la norme Arinc 651 (ref. 11).

L'objectif de la norme Arinc 651 est donc de proposer une architecture de calculateur qui utiliserait des modules standards. L'objectif est, en particulier, de pouvoir différer toute action de maintenance. Cet objectif est quantifié : la probabilité d'avoir à remplacer un module doit être inférieure à 1% pendant 200 heures après la première panne simple d'un calculateur. En première approximation, les commandes de vol de l'A320 sont très proches de tenir cet objectif. L'intérêt d'utiliser des modules standards, remplaceables indépendamment les uns des autres est donc plutôt d'une part de faciliter la maintenance en compagnie, avec en particulier une réduction du stock de rechange, d'autre part de réduire le coût d'achat du système. L'une de nos études en cours tendrait à conserver l'approche A320 d'utilisation de deux types de calculateur. L'un pourrait être de technologie classique, et l'autre se couler dans le moule de la norme Arinc 651.

Une autre conséquence de l'intégration est la possibilité de disposer de circuits répondant à des fonctions spécifiques, et donc faisant l'objet de productions en petite série (Application Specific Integrated Circuit). Ceci permet d'envisager de construire des circuits spécifiques de la tolérance aux

fautes, comme un voteur reconfigurable (tel que celui défini en ref. 13), ou encore un chien de garde évolué, permettant de surveiller finement le déroulement d'un programme. Un tel chien de garde est développé par l'Aérospatiale pour un calculateur de commandes de vol de l'A340.

La manière de programmer les calculateurs est également en cours d'évolution. Il est vraisemblable que l'usage d'ADA va se répandre. Ce langage sera utilisé sur l'A340 pour un calculateur de commandes de vol.

### 2.2.2. Système distribué de commandes de vol

L'utilisation de bus numériques à accès multiplexé peut être transparente ; sans impact notable sur le système. Néanmoins, un accompagnement de l'introduction de bus multiplexés peut être une délocalisation de calculateurs de commandes de vol, et une remise en cause de l'architecture du système. La délocalisation de calculateurs peut conduire à en disséminer dans la soute, voire à en placer (en tout ou partie) à proximité des actionneurs (ce qui est fait maintenant de façon courante pour la commande des moteurs). Cette dernière approche est utilisée sur l'A320 pour des électroniques d'asservissement, et sur l'A340 : le calculateur numérique chargé de l'asservissement du Plan Horizontal Réglable est placé à proximité de celui-ci.

### 2.2.3. Commandes de vol optiques

Un système de commandes de vol peut être composé d'un sous-système construit autour de calculateurs qui assure le fonctionnement normal du système, et d'un sous-système de secours. Ce sous-système peut être à transmission mécanique, ou construit autour de chaines de calcul numériques, ou analogiques avec des transmissions électriques ou optiques. Un effort important est mené à l'Aérospatiale dans l'étude de la dernière solution : transmission optique (voir ref. 14).

Les principes de base de cette architecture sont d'utiliser une source d'énergie primaire hydraulique, et de placer tous les composants électroniques et les cables électriques associés dans une enceinte blindée, voire de les intégrer à l'actionneur (voir figure 6). Ce calculateur peut asservir une servo-commande en fonction de la position du manche, et éventuellement d'un capteur inertiel (gyromètre par exemple). La position du manche est mesurée au moyen d'un capteur optique (voir ref. 15). Ce capteur est passif, et quatre fibres optiques sont utilisées. La position est fonction de la différence entre ce que le calculateur a émis sur une fibre, et ce qu'il a reçu sur l'autre, modifié par le capteur. Le capteur inertiel est placé dans la même enceinte que le calculateur. L'énergie électrique nécessaire au calculateur est fournie par conversion d'énergie hydraulique via une micro-génération. Dans l'étude actuelle, la servo-commande est utilisée soit par un des calculateurs du système de base, soit par le calculateur de secours. L'interface envisagé est que chaque calculateur ait ses propres capteurs, et que la servo-valve qui permet de commander la servo-commande soit à deux enroulements, chaque calculateur utilise un des deux enroulements. En régime permanent, un seul des deux calculateurs asservit l'actionneur. Le principe de la commutation est que le calculateur digital émet en permanence (via une fibre optique) des messages vers le calculateur de secours. Si cette émission est interrompue, le calculateur de secours asservit l'actionneur. A ce jour, chacun des composants a été testé séparément, les composants de technologie optique (capteur, transmission) sont testés en vol, et les tests d'intégration ont été effectués. Ces divers tests ont permis de montrer la viabilité du concept.

## 2.3. Evolution des méthodes - atelier système

Le développement d'un système tel que celui des commandes de vol suit un cycle de développement (voir figure 7) qui part des besoins exprimés au niveau avion. Un ensemble de méthodes et outils a été développé à l'Aérospatiale pour effectuer au mieux ce développement. Cet ensemble constitue un atelier de conception de systèmes, ou atelier système. Cet atelier n'est encore qu'une collection d'outils plus ou moins interconnectés, supportés par des ordinateurs parfois incompatibles, ayant parfois été conçus pour un système particulier. L'effort actuel continu de porter sur les outils eux-mêmes, mais a également pour but de rapprocher ses outils entre-eux, de manière à disposer d'un ensemble structuré d'outils communicants.

Il est également nécessaire d'essayer de prévoir les outils susceptibles d'être utiles dans

le futur. En particulier, l'émergence de systèmes de commandes de vol distribués est possible. Ceci peut entraîner des problèmes de synchronisation plus difficiles à maîtriser que dans la situation actuelle. Les réseaux de Petri sont donc explorés en avance pour éventuellement traiter ce genre de problème (voir en annexe).

Les méthodes utilisées doivent permettre de valider une conception. Elles peuvent influer également sur la méthode de conception pour que celle-ci soit validable. Ces deux concepts (validation, et conception pour la validation) sont essentiels à la conception d'un système critique tel que les commandes de vol. Ils apparaissent particulièrement dans le choix du langage de spécification utilisé. Un point pivot dans le cycle de conception d'un système est l'écriture de la spécification fonctionnelle des équipements. Cette spécification est structurée suivant une décomposition fonctionnelle en livres/chapitres/planches. Pour ce faire, un langage de spécification a été développé à l'Aérospatiale. Ce langage dit langage SAO (Spécification Assisté par Ordinateur) comprend des opérateurs et des règles syntaxiques de combinaisons de ces opérateurs. Ces opérateurs sont particulièrement adaptés à la spécification d'équipements de commandes de vol car il utilise les symboles de base de l'automatique et de la logique. Le langage comporte également des symboles adaptés à d'autres systèmes comme le système de gestion des alarmes ou des informations affichées au pilote. Le langage permet donc de transcrire assez directement les études de définition du système en spécification d'équipements. Ce langage a une définition formelle, ce qui limite les risques d'ambiguïté et d'incohérence. Ce langage est supporté par un outil de saisie graphique. La validation de la spécification est facilitée par le langage choisi. D'une part, il permet d'inclure des points de piquage d'information pour faciliter le dépouillement des essais au sol ou en vol, d'autre part, il permet l'utilisation d'outils de vérification et de validation de spécification. Ces derniers outils sont décrits plus particulièrement § 2.3.2.

La gestion des relations entre les équipes de développement et les services de production, gestion, et après-vente est également assurée par l'atelier système. Sont ainsi gérés ou en passe de l'être la liasse électrique, les équipements, les demandes d'évolution d'équipements, la description du système.

La conception d'un avion civil est constitué d'activités qui toutes ont la sécurité de l'avion soit comme objectif, soit comme contrainte. Tous les outils de l'atelier système sont donc liés à des degrés divers à la sécurité de l'avion. Les outils plus particulièrement liés à la sécurité sont décrits dans la suite.

### 2.3.1. Définition du système des commandes de vol

La définition du système demande à attribuer à chaque gouverne un certain nombre d'actionneurs, et pour chaque actionneur une source d'énergie et des calculateurs. L'écriture d'un tel arrangement implique de vérifier que les objectifs de sécurité du système sont tenus. Il est alors nécessaire d'envisager un nombre important de combinaisons de pannes, ce nombre pouvant être de quelques milliers. Une étude a été menée, visant à automatiser ce processus.

Il s'est avéré d'une part utile de disposer d'un outil permettant d'évaluer un grand nombre de cas de pannes, permettant l'utilisation de fonctions de capacité (voir ref. 16), et d'autre part que la possibilité de pouvoir modéliser des dépendances statistiques n'était pas absolument nécessaire, quite à parfois fournir un résultat pessimiste. Cette étude a aboutie à un outil informatique qui est utilisé actuellement à la définition des nouveaux avions Airbus (A340, A330). Cet outil (appelé VERIFCDVE, contraction de "vérification", et de "commandes de vol électriques") prend en entrée un arrangement de calculateurs, d'actionneurs, de sources d'énergie hydraulique et électrique, mais également d'événements particuliers tels que l'arrêt simultané de tous les moteurs, et donc d'un grand nombre de sources d'énergie. La disponibilité d'une surface est fonction de la disponibilité de certaines de ces ressources. Cette description est faite avec un support du type arbre de fautes.

La fonction de capacité utilisée permet de définir la manoeuvrabilité en roulis de l'avion, en fonction de l'état de dégradation du système des commandes de vol. Cette manoeuvrabilité peut être approchée par la fonction suivante qui mesure le taux de roulis disponible par une fonction linéaire des surfaces disponibles :

$$\sum_{G \in \{\text{gouvernes disponibles}\}} (\text{taux de roulis de la gouverne } G)$$

En définissant un seuil d'acceptabilité

((taux de roulis > X) => (manoeuvrabilité suffisante))

il est alors possible de diviser les états de dégradation du système en états de succès ou d'échec, et ainsi de calculer la probabilité de défaillance du système par rapport à l'objectif de manoeuvrabilité en roulis.

L'outil crée automatiquement des combinaisons de pannes et évalue la disponibilité des surfaces, et donc une fonction de manoeuvrabilité en roulis. Il compare ces résultats à des objectifs. Ces objectifs sont d'une part de manoeuvrabilité (disponibilité des gouvernes de profondeur, taux de roulis disponible, etc) et d'autre part fiabiliste (un objectif de manoeuvrabilité doit être tenu pour toute combinaison de pannes dont la probabilité est supérieure à un objectif donné de fiabilité). L'outil liste alors les combinaisons de pannes qui ne tiennent pas les objectifs (s'il y en a), et donne pour chaque objectif de manoeuvrabilité, la probabilité de non satisfaction. L'outil prend également en compte les possibilités d'impasse technique (par exemple le décollage avec un calculateur en panne).

Cet outil s'est révélé particulièrement utile dans la phase de conception de l'architecture du système des commandes de vol de l'A340. En effet, beaucoup d'architectures ont été envisagées, et chacune d'elles devait être vérifiée. Cette vérification peut se faire sans outil, mais c'est un travail fastidieux (typiquement, des milliers de combinaisons de pannes sont à prendre en compte pour faire une vérification vraiment fine), et donc avec des risques d'erreur. L'outil a donc permis d'étudier plus d'architectures, donc d'avoir un produit final de meilleure qualité (en terme de nombre d'asservissements par calculateur, donc en terme de masse, de puissance de calcul, et de coût), et globalement de gagner du temps. Un nouvel outil est actuellement en cours de réalisation, utilisant des techniques de système expert. Son objectif est double. D'une part, il doit pouvoir affiner la représentation du système (prise en compte des logiques de reconfiguration définies dans les spécifications fonctionnelles, raffinement de la fonction de capacité), d'autre part, il doit pouvoir évaluer des systèmes autres que les commandes de vol (le système des instruments de vol par exemple).

### 2.3.2. Vérification et validation des spécifications fonctionnelles

Certaines activités de vérification des spécifications fonctionnelles sont supportés par des outils informatiques. Ainsi, la syntaxe de la spécification peut-elle être vérifiée automatiquement. Un outil de gestion de configuration est également disponible et utilisé.

La validation de la spécification est faite principalement par relecture (en particulier lors de l'analyse de sécurité) et par les tests au sol (voir ref. 17) ou en vol. De plus notre objectif est une validation au plus tôt. Pour ce faire, divers outils de simulation existent, et ce grace au fait que les spécifications sont écrites dans un langage formel, qui rend la spécification exécutable. Il est ainsi possible de simuler une partie d'une spécification (outil LIS), ou encore (outil OSIME) l'ensemble du système des commandes de vol (calculateurs, actionneurs, capteurs, retours avion). En outre, la partie de spécification qui décrit les lois de pilotage peut être simulée en temps réel (outil OCAS), en prenant ses entrées d'un manche latéral réel (en fait plus simple qu'un manche avion). Les scénarii de tests ainsi générés peuvent être enregistrés et rejoués ultérieurement, sur une version suivante de la spécification par exemple. Ceci permet de faire un test de non régression. Les signaux à observer peuvent être choisis arbitrairement, et ne sont pas limités aux entrées/sorties d'une planche de spécification. Les outils OSIME et OCAS sont couplés à un modèle aérodynamique de l'avion.

### 2.3.3. Validation de la sûreté de fonctionnement

L'analyse de sécurité d'un système aussi complexe que les commandes de vol est un processus difficile à mettre en oeuvre. Pour simplifier cette lourde tâche, l'Aérospatiale prépare un outil d'aide à la gestion de cette analyse. L'ensemble des outils qui sont développés le sont sous le terme générique de RAMS-ES/A (Reliability Availability Maintenability Safety - EnvironmentS / Aircraft). Dans un premier temps, les fonctions suivantes vont être automatisées : 1) support à l'analyse de sécurité, l'utilisateur n'aura qu'à fournir l'information nécessaire à l'analyse, la présentation et la mise en page étant gérées automatiquement, avec une vérification de la cohérence de l'information (outil SARA), 2) gestion des informations de l'analyse de sécurité d'un système (génération électrique par

exemple) qui sont utiles pour une autre analyse (commandes de vol par exemple), 3) élaboration de synthèses au niveau avion de toutes les analyses de sécurité (outil DAISY associant tous les systèmes, en particulier la génération électrique et les commandes de vol de l'exemple précédent). Une deuxième étape permettra d'automatiser en partie l'écriture du manuel de maintenance (les intervalles entre inspections qui sont indiqués dans le manuel de maintenance seront extraits automatiquement des analyses de sécurité), de disposer d'une banque de données basée sur l'expérience acquise sur d'autres avions, d'étudier facilement, après la mise en service d'un avion, l'impact sur les objectifs de sécurité de la fiabilité des équipements mesurée en exploitation réelle.

Il est également important de surveiller la qualité des logiciels embarqués. L'Aérospatiale, en tant qu'avionneur, effectue de nombreux audits chez ses fournisseurs de logiciels. Une méthode d'audit et un guide d'audit ont été développés. Ces moyens permettent de faire un point sur l'état d'un logiciel du point de vue qualité, mais aussi d'étudier les tendances de cette qualité. Ces moyens sont maintenant en grande partie informatisés.

### 2.3.4. Programmation automatique

La programmation automatique n'est pas une activité de conception système. Elle est néanmoins étroitement mêlée dans la mesure ou la programmation automatique se fait à partir de la spécification fonctionnelle qui est elle un des produits de l'activité de conception du système.
L'utilisation d'outils de programmation automatique tend à se généraliser. Cette tendance est apparue sur l'A320 et se confirme sur l'A340 (en particulier, deux calculateurs de commandes de vol seront en partie programmés automatiquement). L'utilisation de tels outils à un impact positif sur la sécurité. Un outil automatique permet d'assurer qu'une modification de spécification sera codée sans "stress", même si cette modification est à faire rapidement (situation rencontrée lors de la phase d'essai en vol par exemple). De plus, la programmation automatique, au travers de l'utilisation d'un langage formel de spécification, permet de réutiliser d'un programme avion à l'autre du code embarqué. Il est à noter que les outils de validation de spécification fonctionnelle (§ 2.3.2) utilisent un outil de programmation automatique. Cet outil présente des parties communes avec l'outil de programmation automatique utilisé pour la génération de code pour les calculateurs de commandes de vol. Ceci accroit la puissance de validation des simulations.

## 2.4. Intégration d'une évolution - cas du manche latéral

Les premiers essais d'un manche latéral et d'une loi de pilotage en profondeur du type de celle qui sera plus tard utilisée sur A320 ont eu lieu sur Concorde en 1978, soit dix ans avant la mise en service commercial de l'A320. En 1983, un Airbus A300 a servi de banc d'essai volant. L'avion était équipé d'un manche latéral en place gauche pour laquelle les organes de pilotage classiques avaient été supprimés. 75 heures de vol ont été effectuées, avec 48 pilotes des Services Officiels, d'Airbus Industrie, et de compagnies aériennes. Ces vols n'ont fait apparaître aucune difficulté d'adaptation au manche latéral. Un accord général s'est fait sur la loi de pilotage en profondeur, ainsi que sur les protections du domaine de vol, en particulier pour la protection haute incidence. Il est par contre apparu nécessaire d'améliorer la loi de pilotage en latéral. Avant les essais en vol de l'A320 (1987), une nouvelle campagne d'essai sur A300 a permis de valider l'utilisation d'un manche latéral à gauche comme à droite, et d'affiner les lois de pilotage qui devaient être utilisés sur l'A320.

En parallèle à ces études, un modèle de charge de travail de l'équipage (ref. 18) a été mis au point dans l'optique de la certification du premier A300 avec un cockpit de concept "tout à l'avant" (1982). Ce modèle a été affiné lors d'essai sur les avions qui ont suivi (A310, A320), et a servi à valider le concept de manche latéral et des automatismes associés.

## Annexe

Les réseaux de Petri ont un intérêt théorique reconnu. Il est néanmoins nécessaire de valider cette approche sur un cas concret. L'exemple choisi est un système des commandes de vol distribué. Un système est dit "distribué" dans la mesure ou un réseau de communication existe, et que tout abonné est autonome, qu'aucun n'est indispensable, et que l'ensemble doit coopérer pour mener à bien la tâche qui est confiée au système.Diverses études ont été menées sur l'architecture d'un système

de commandes de vol, tant dans l'industrie que la recherche (ref. 19, 20, 21, 22). Une étude à été menée à l'Aérospatiale, d'une part pour définir un système de commandes de vol distribué, d'autre part pour mettre sur pied une méthode de validation d'un tel système.

Une approche saine est de concevoir le système en couche, une couche de niveau inférieur assurant un service à la couche immédiatement au dessus. Les couches envisagées sont similaires à celles utilisées dans un système distribué tolérants aux fautes développé à l'Université de Californie, Los Angeles (DEDIX, ref. 20). Ces couches sont représentées sur la figure 8. La couche dite "transport" serait définie par la norme Arinc 629. Cette couche assure les échanges de données entre calculateurs, et assure qu'un message émis est bien reçu (une éventuelle attaque de généraux byzantins devrait être traitée à ce niveau). La couche de synchronisation a pour objet de collecter les données qui doivent être traitées en même temps. La couche de tolérance aux fautes masque les redondances à la couche qui contient le logiciel d'application. Par exemple (figure 8), soit un système de commandes de vol relié à trois centrales à inertie. Chaque centrale va périodiquement émettre la vitesse de tangage qu'elle a mesurée. Cette émission et la réception dans le calculateur de commandes de vol sont gérées par la couche transport. Les trois vitesses de tangage émises sont regroupées par une tâche appartenant à la couche de synchronisation. Un vote est effectué dans la couche dite de tolérance aux fautes, pour masquer une éventuelle erreur d'une des trois centrales. Le résultat du vote est utilisé par l'application : les lois de pilotage. Un schéma identique pourrait être appliqué à la synchronisation d'intégrateurs.

L'objectif de l'activité de synchronisation des calculateurs n'a pas pour objet de synchroniser les horloges, mais plutôt de synchroniser les données. Typiquement, les calculateurs sont redondants et leurs sorties sont comparées ou votées pour détecter la panne de l'un d'entre-eux. Si les calculs sont effectuées à partir de données (trop) différentes, un risque de divergence existe et donc de déconnexion intempestive. Une synchronisation est donc nécessaire. Par contre, une synchronisation très stricte, au niveau horloge en particulier, condamne l'utilisation de logiciels dissimilaires, et serait contraire à notre pratique. Nous nous sommes donc orientés pour cette étude sur une synchronisation de données, ce qui implique du point de vue temporel une syncronisation "lâche".

Un protocole de synchronisation lâche, de données, a été conçu dans l'optique d'une utilisation dans un système distribué de commandes de vol. Il s'apparente au traitement qui est fait sur les avions précédents de certaines données. Il s'appliquerait bien à un ensemble de calculateurs devant *émettre des consignes vers des actionneurs, et, pour éviter* de diverger, devant prendre des données d'entrée de valeur sensiblement égale. La base du protocole est que les calculateurs ont un fonctionnement cyclique, et périodiquement vont émettre des consignes vers les actionneurs et les données à synchroniser. Cet ensemble de données est un message unique au niveau de la couche d'application. Ayant émis son message, le calculateur va attendre les messages de tous les autres calculateurs avec lesquels il doit se synchroniser. Ces messages ayant été reçus, le calcul des lois de pilotage peut reprendre à partir de la moyenne ou de la médiane (ou autre) des données à synchroniser. Ce schéma idéal peut être mis en défaut par la panne d'un calculateur qui n'émettrait plus. Le système ne doit pas être bloqué par ce cas de panne. Tant par principe que pour couvrir un cas de point commun entraînant une défaillance simultanée de plusieurs calculateurs, le protocole doit survivre à l'isolement du calculateur qui l'exécute. Nous avons donc trois modes de fonctionnement du protocole

- fonctionnement normal de tous les calculateurs (figure 9.a),
- fonctionnement normal d'une majorité de calculateur (figure 9.b),
- isolement d'un calculateur (figure 9.c).

En fonctionnement normal de tous les calculateurs, ceux-ci émettent leur message de synchronisation quasiment en même temps. L'étape de synchronisation est terminée dès réception de tous les messages.

En fonctionnement normal d'une majorité de calculateur (figure 9.b), seul un petit nombre de calculateurs est hors d'état d'émettre un message. Un chien de garde est armé dès réception du message qui, ajouté aux messages précédemment reçus, permet d'affirmer qu'une majorité de calculateurs est en état d'émettre et a émis. Si "n" calculateurs sont à synchroniser, ce chien de garde est armé dès réception du "$m^{ième}$" message, avec (n = 2m-1, n impair), ou (n=2m-2, n pair). Dans les deux cas, n et m vérifient (2m>n), et donc aussi le fait majoritaire. Pour éviter qu'un calculateur isolé ne soit bloqué un second chien de garde doit également être armé.

Nous venons de décrire informellement le protocole de synchronisation. L'étape suivante est de le spécifier de façon formelle, et que cette spécification soit validable. Pour ce faire, le protocole

a été spécifié en utilisant les réseaux de Petri. Le réseau résultant apparaît sur la figure 10. Ce réseau peut être découpé en un bloc d'acquisition de messages valides (P1, P2, P4, T1, T2, T3), un bloc de détection de la fin de la phase de synchronisation (tous les messages sont reçus, ou déclanchement d'un chien de garde, P4, T4, T5, T6), un bloc de fin de synchronisation, vote, réinitialisation du protocole (P7, P8, T7, T8, T9), un bloc d'interaction via la couche de transport avec les autres calculateurs (P9, T1), l'application (P11 à P15, T10 à T15).

L'utilisation des réseaux de Petri a été motivée par la nécessité de disposer d'un moyen de description/spécification d'un protocole de synchronisation, l'aspect formel de ce moyen étant un point important pour éviter toute ambiguïté. Une autre motivation tout aussi importante est la nécessité de pouvoir valider le protocole. Ceci a été réalisé de deux voies complémentaires, d'une part en utilisant les propriétées intrinsèques des réseaux de Petri qui permettent de dégager des "invariants" du protocole modélisé, d'autre part en simulant le fonctionnement du protocole. Il est à noter que cette phase de validation a été supportée par un outil informatique : RdPS (ref. 23).

Un invariant d'un réseau de Petri traduit une propriété du réseau et en apporte une preuve formelle. Par exemple, en notant M(Pi) le nombre de jeton sur la place Pi (son marquage), une analyse statique du réseau fait apparaître l'existence de l'invariant de place suivant :

M(P7) + M(P8) + M(P9) = 1

Ceci traduit le fait qu'il y a exclusion mutuelle entre la prise en compte du médium (P9), la tâche de vote (P7), et la terminaison de la synchronisation (P8). Ceci signifie en particulier qu'un message qui arriverait pendant l'exécution des tâches de vote ou de terminaison de la synchronisation serait ignoré temporairement, et pris en compte au round de synchronisation suivant. L'examen des invariants ne permet pas de garantir complètement la validité du protocole. Il permet néanmoins d'exhiber des propriétés du protocole, qui sont acceptables ou non. En outre, si toute place appartient à au moins un invariant, il est possible d'affirmer que le réseau est borné, et donc l'absence de boucle infinie.

La simulation du réseau permet de générer son graphe de marquage, c'est-à-dire l'ensemble des états que peut prendre le protocole. Trois types de vérification peuvent être effectués. Tout d'abord, une analyse automatique permet de vérifier que le réseau est vivant et que le protocole ne peut donc pas se bloquer. Une autre vérification est d'examiner tous ces états, et les passages d'un état à l'autre pour apprécier si ce comportement du protocole ainsi décrit correspond à ce qui était attendu. Cette vérification est utile mais peut s'avérer fastidieuse si faite sans méthode. Dans le cas du protocole étudié, le nombre d'état est fonction du nombre de calculateurs ("n") interconnectés. La vérification doit donc se faire pour le nombre maximal de calculateurs, mais aussi pour tous les états possibles de dégradation, soit avec n-1 calculateurs, ainsi qu'avec n-2 seulement, jusqu'à la disponibilité d'un seul calculateur. Une récurrence apparaît à l'examen et il est possible de créer une grille de lecture valable quel que soit le nombre de calculateurs disponibles. Enfin, il est possible de faire une recherche automatique de certains états, qui pourrait être indésirable. Cette recherche peut servir à confirmer l'examen du graphe de marquage, pour les points touchant à la sécurité. Par exemple, il peut être estimé que la tâche de vote ne peut pas être active en même temps que le calcul des lois (tâche d'application) qui est censé utiliser le résultat de ce vote. Si le protocole pouvait entrer dans un tel état, un marquage tel que M(P7) = M(P11) = 1 existerait. Ce type de marquage peut être recherché automatiquement.

En règle générale, et pour un système de commandes de vol en particulier, il est nécessaire d'associer à toute activité de conception une activité de validation. Il est aussi nécessaire de concevoir avec la validation comme objectif. Notre approche vis-à-vis des systèmes distribués est donc autant de méthode (les réseaux de Petri en sont une) que de technique de transmission.

Dans la même voie, les réseaux de Petri stochastiques (voir ref. 23) peuvent ajouter une dimension fiabiliste à la modélisation d'un système. Une première étude a permis de définir des besoins en matière d'outils informatiques, ce qui a conduit à une étude des implications théoriques et pratiques de ces besoins par le Centre National des Arts et Métiers de Paris (ref. 24), et à la réalisation d'un prototype.

**Références**

1 : B. Ziegler, et M. Durandeau, "Flight control system on modern civil aircraft", publié dans les actes de "International Council of the Aeronautical Sciences - ICAS84", septembre 1984, Toulouse.
2 : X. Paris, "Control laws of the A320 Airbus", à paraître dans "Concise Encyclopedia of Aeronautics

ans Space Systems", Pergamon Press.
3 : M. Durandeau, et J. Troyes, "Les commandes de vol électriques des avions de transport", publié par le "Cercle des officiers mécaniciens et ingénieurs navigants de l'aviation civile - COMINAC", janvier 1986, Roissy en France.
4 : J. Farineau, "Lateral electric flight control laws of the A320 based upon eigenstructure assignment technique", publié dans les actes du "AIAA Guidance, Navigation and Control Conference", Boston, aout 1988.
5 : H. Lansdorf, "Terminal weather", publié dans "Flight International", 23 mai 1987, pp.44-48.
6 : "Software considerations in airborne systems and equipment certification", publié par "Radio technical commission for aeronautics" (RTCA) et par "European organization for civil aviation electronics" (EUROCAE), N° DO178A/ED12A, Mars 1985.
7 : P. Traverse, "Sûreté des systèmes informatiques embarqués à bord d'avions", publié dans les actes du "3ième Colloque International sur la Sécurité Aérienne et Spatiale", Toulouse, septembre 1988.
8 : A. Postlethwaite, "Tupolev's new twin", Flight International, 20 mai 1989, pp. 44-46.
9 : A. Postlethwaite, "Ilyushin goes the distance", Flight International, 20 mai 1989, pp. 49-51.
10 : R.J. Bleeg, "Commercial jet transport fly-by-wire architecture considerations", dans les actes de "8th AIAA/IEEE Guidance and Control Conference", 1988, pp. 399-406.
11 : Aeronautical Radio, INC, "Design guidance for integrated modular avionics", AEEC letter 89-053/SAI-357, Mai 1989.
12 : Aeronautical Radio, INC, "Multi-transmitter Data Bus, part 1, Technical Description".
13 : J. Grossin, et P. Traverse, "Système de commandes de vol pour aéronef", brevet français N° 88.03.343, mars 1988.
14 : J.P. Laborie, P. Desjean, J.P. Domergue, et P. Palandjian, "Système pour la commande d'une surface aérodynamique mobile d'un aéronef", brevet français N° 86.01.576, février 1986.
15 : J.P. Domergue, "A fibre optic moving part position determination by variable metallized sheet optical elements whose displacementinfluences relative outputs of two photo-detectors", brevet européen N° EP 190181 A.
16 : D. Beaudry, "Performance-Related Reliability Measures for Computing Systems", publié dans "IEEE Transactions on Computers", Vol. C-27, N°6, Juin 1978, pp. 540-547.
17 : D. Chatrenet, "Simulateurs A320 d'Aérospatiale : leur contribution à la conception, au développement et à la certification", publié dans les actes d'INFAUTOM 89, Toulouse, mars 1989.
18 : R.D. Blomberg, A.L. Schwartz, J.J. Speyer, et J.P. Fouillot, "Application of the Airbus workload model to the study of errors and automation", publié dans les actes du "3ième Colloque International sur la Sécurité Aérienne et Spatiale", Toulouse, septembre 1988.
19 : J.M. Wensley, L. Lamport, J. Goldberg, M.W. Green, K.N. Levitt, et P.M. Melliar-Smith, "SIFT: The Design and Analysis of a Fault-Tolerant Computer for Aircraft Control", publié dans les "Proceedings of the IEEE", Vol. 66, N°10, octobre 1978, pp.1255-1268.
20 : A. Avizienis, P. Gunningberg, J.P.J. Kelly, L. Strigini, P.J. Traverse, K.S. Tso, et U. Voges, "The UCLA DEDIX System: A Distributed Testbed for Multiple-Version Software", publié dans les actes du "15th International Symposium on Fault-Tolerant Computing - FTCS15", Juin 1985, Ann Arbor, Michigan, pp.126-134.
21 : Ch. Hourtolle, "Conception de logiciels sûrs de fonctionnement : Analyse de la sécurité des logiciels ; Mécanismes de décision pour la programmation en N-versions", thèse de doctorat de l'Institut National Polytechnique de Toulouse, N°122, octobre 1987.
22 : D.P. Glutch, et M.J. Paul, "Fault-Tolerance in Distributed Digital Fly-by-Wire Flight Control Systems", publié dans les actes du "7th Digital Avionics Systems Conference - DASC", octobre 1986, Fort Worth, Texas.
23 : G. Florin, et S. Natkin, "Les Réseaux de Petri Stochastiques", publié dans "Technique et Science Informatiques", vol. 4, N°1, 1985, pp.143-160.
24 : K. Barkaoui, G. Florin, C. Fraize, B. Lemaire, et S. Natkin, "Reliability Analysis of Non Repairable Systems Using Stochastic Petri Nets", publié dans les actes du "18th International Symposium on Fault-Tolerant Computing - FTCS18", Juin 1988, Tokyo.

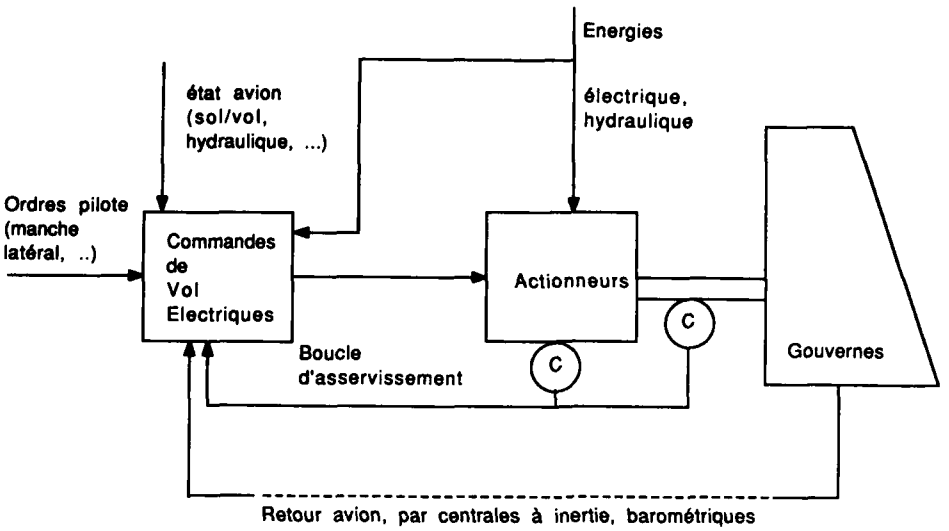



**figure 1 : principe des commandes de vol électriques**

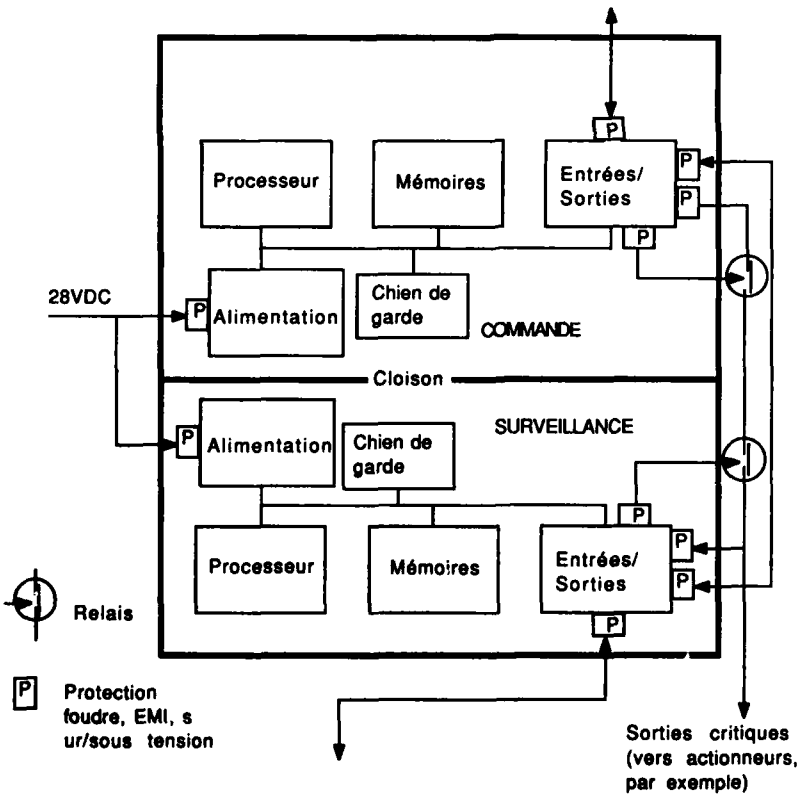



**figure 2 : calculateur à commande et surveillance**

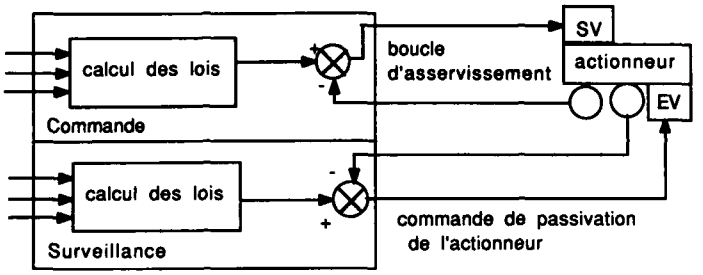



figure 3 : détection d'erreur

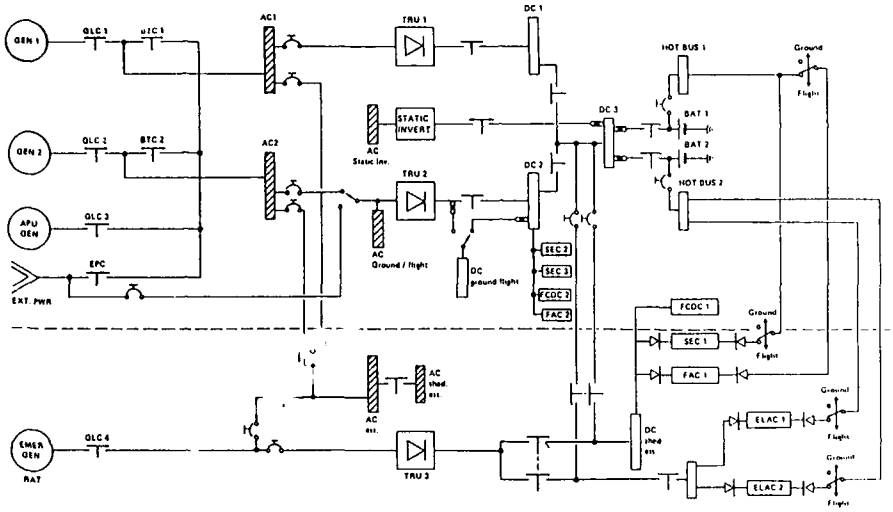



figure 4 : génération et distribution électrique

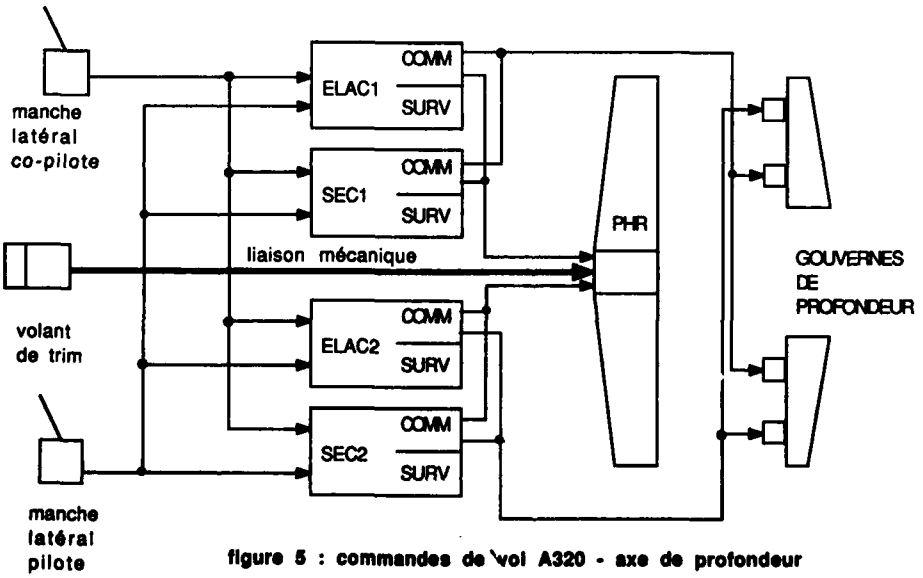



figure 5 : commandes de vol A320 - axe de profondeur

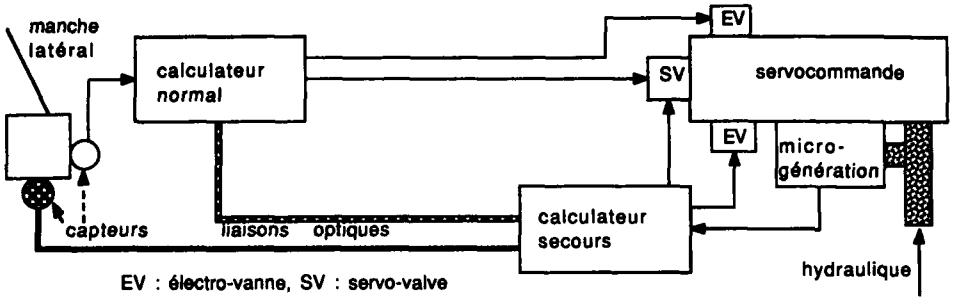



**figure 6 : commandes de vol optiques**

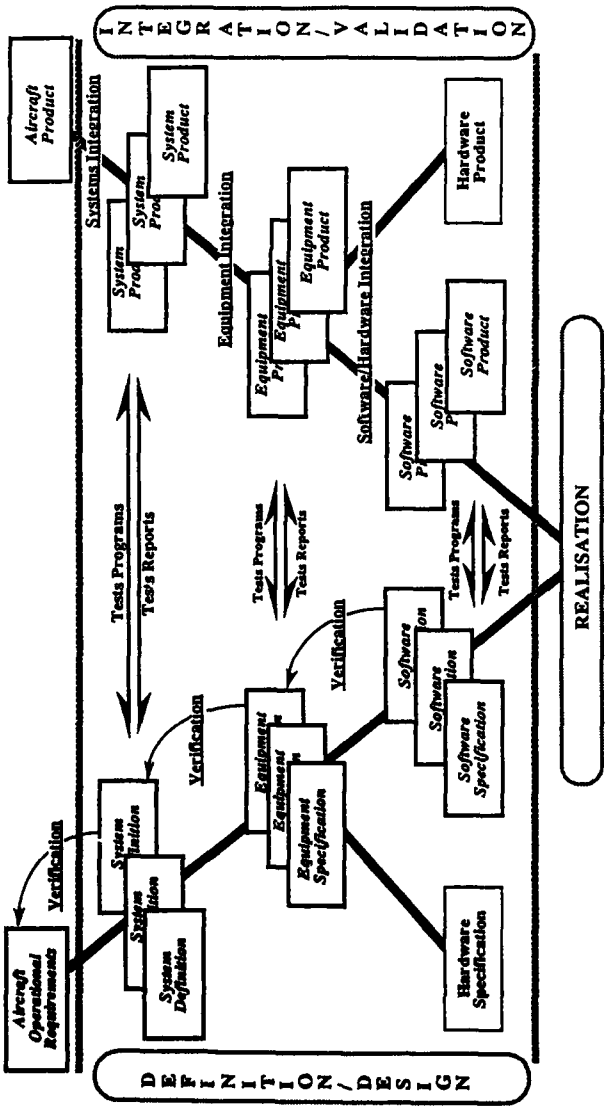



**figure 7 : cycle de développement système**

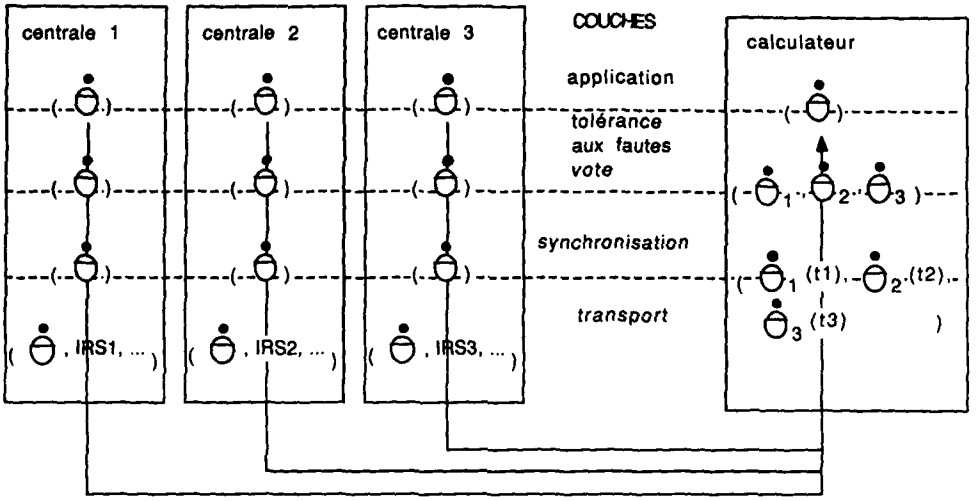



**figure 8 : flot de données**

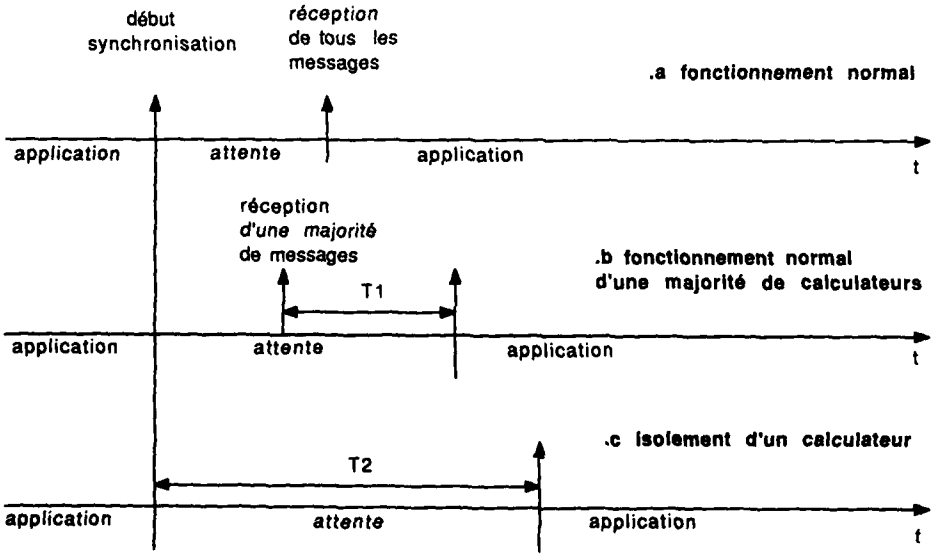
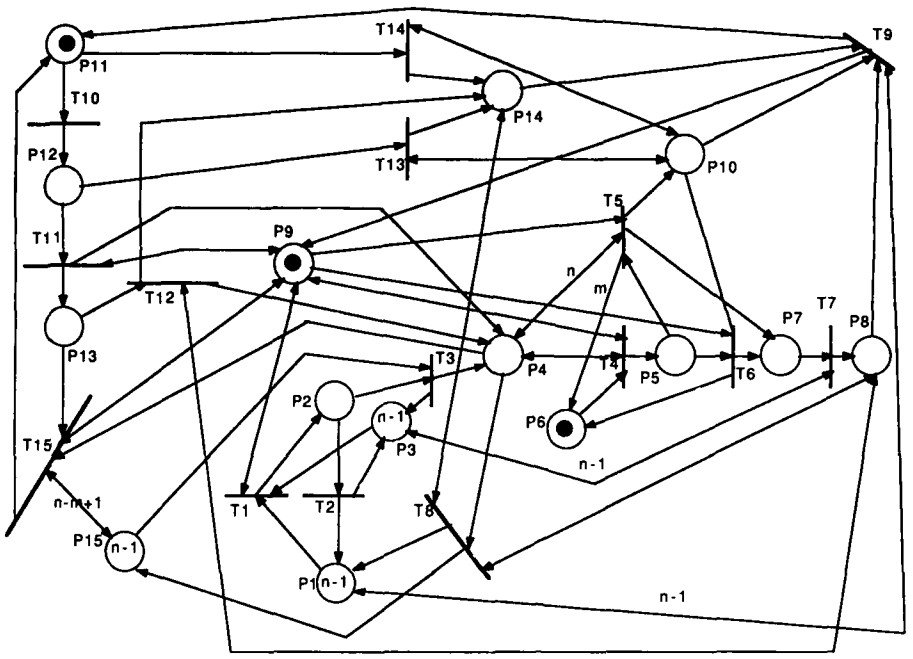



**figure 9 : modes de fonctionnement**




PLACES

| | |
|---|---|
| P1 | nombre maximum de messages |
| P2 | vérification syntaxique des messages |
| P3 | place complémentaire de P2 |
| P4 | stockage des messages reçus et corrects |
| P5 | armement d'un chien de garde, pour détecter les calculateurs qui n'émettent plus |
| P6 | place complémentaire de P5 |
| P7 | vote |
| P8 | fin de la synchronisation |
| P9 | prise en compte du médium permise |
| P10 | place complémentaire de P9 |
| P11 | exécution de la tâche d'application |
| P12 | en attente d'envoi de message de synchronisation |
| P13 | en attente de fin de synchronisation et armement d'un chien de garde pour détecter l'isolement du calculateur |
| P14 | attente de la fin de la synchronisation |
| P15 | nombre de messages non encore acceptés |

TRANSITIONS

| | |
|---|---|
| T1 | lecture d'un message sur le médium |
| T2 | un message reçu est incorrect |
| T3 | un message reçu est correct |
| T4 | instantanée |
| T5 | instantanée |
| T6 | déclanchement du chien de garde armé en P5 |
| T7 | fin du vote |
| T8 | instantanée |
| T9 | instantanée |
| T10 | la tâche est prête à se synchroniser |
| T11 | envoi d'un message |
| T12 | instantanée |
| T13 | le calculateur est en retard, l'assume, n'émet pas, et cherche à se synchroniser sur les autres calculateurs |
| T14 | le calculateur est en retard et est interrompu |
| T15 | déclanchement du chien de garde armé en P13 |

**figure 10 : spécification du protocole**

# PILOT MONITORING OF DISPLAY ENHANCEMENTS GENERATED FROM A DIGITAL DATA BASE

by

**Peter J.Bennett and John J.Cockburn**
Ferranti Defence Systems Limited
1 South Gyle Crescent
Edinburgh EH12 9HQ
United Kingdom

## SUMMARY

A Ferranti International integrated covert mission system called PENETRATE (Passive Enhanced Navigation with Terrain Referenced Avionics) is currently undergoing flight trials on a Hunter fast-jet aircraft at the Royal Aerospace Establishment, Farnborough, England. The heart of the PENETRATE system is a digital data store housing a three dimensional model of the terrain including cultural details and tactical intelligence information. This integrated mass memory store supplies data to a Terrain Referenced Navigation System, a head-down Digital Map and a head-up Skeletal Perspective Terrain Image Generator. The integrity of the terrain data loaded into this covert system cannot be totally guaranteed; neither can the navigation accuracy. The pilot must, therefore, use his normal visual technique to monitor the synthetic terrain displays for acceptable correlation with the real world.

This paper describes the PENETRATE integrated covert mission system, the increase in operational capability it provides and the visual monitoring requirements.

## INTRODUCTION

Electro-optic sensors such as Forward Looking Infra Red (FLIR) and Night Vision Goggles (NVG) enable aircraft to be flown at high speed and low level in poor visibility and at night. The combination of these passive sensors enables a tremendous increase in operational capability, but they are not the complete solution. To survive against todays sophisticated defences aircrew must make maximum use of stealth penetration techniques and facilities. Even when using both FLIR and NVG sensors, foreground undulations or ridges often lack contrast and are difficult to identify. Power lines and masts also often lack both thermal and visual contrast against the background scene and neither sensor can always be relied upon to pick up these obstructions at a safe avoidance range. As the weather deteriorates and the performance of these electro-optic sensors decreases, additional enhancements are required to continue the mission safely. It is this crucial requirement that the PENETRATE system addresses.

## THE PENETRATE SYSTEM

The PENETRATE system is designed to provide aircrew with extremely accurate navigation coupled with head-up and head-down displays of the terrain. The integrated airborne system comprises a mass data store and three main airborne modules:-

Terrain Referenced Navigation (TRN)
Digital Map Generator (DMG)
Skeletal Perspective Terrain Image Generator

The mass data store is a very large capacity military optical disc drive. This compact store is the heart of the PENETRATE system and contains several layers of information which are accessed by the individual modules, Figure 1.

The first layer is the Digital Terrain Elevation Data (DTED) which is used by all three of the main airborne modules. The next layer is cultural information such as roads, railways, woods and rivers which are required by the digital map. This cultural information can either be based on feature vectors or alternatively it can be obtained by digitizing standard aeronautical charts. Obstructions such as pylons, masts and chimneys are strictly cultural information. They are, however, held as a separate data layer as this information is used by the perspective image generator to display obstruction symbols in the Head-Up-Display (HUD) in order to cue the pilot's attention to these hazards. Intelligence information such as missile sites, lethal zones, and Forward Edge of Battle Area (FEBA) is held in another data layer. This is used by the digital map generator which processes and displays the information in a variety of ways. The final layer of data contains mission information such as the target, waypoints, routeing, timings and fuel bingos. This data is generated using a mission planning system and it is specific to the particular mission being flown. It is generally displayed on the digital map, but certain data such as the target and the planned route can also be processed by the perspective image generator for display on the HUD. Mission specific information can either be held in the data transfer module, which is an adjunct to the mass data store, or it can be written directly on to the optical disk via a data link.

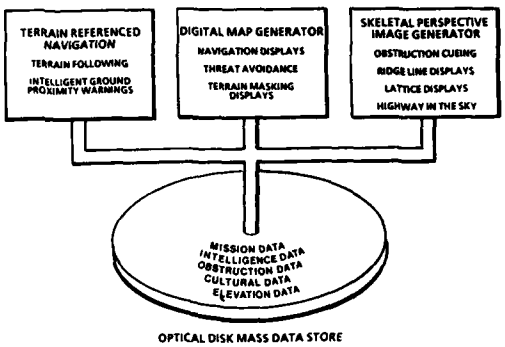



**FIGURE 1**
**MAIN AIRBORNE MODULES OF THE PENETRATE SYSTEM**

For demonstration and development purposes the PENETRATE system has been installed in an avionics pod for carriage on a standard wing pylon of the Nightbird Hunter aircraft at the Royal Aerospace Establishment, Farnborough (Figure 2). Pod inputs consist of electrical power, a serial data bus for inertial parameters, an analogue input for radar height, and a few discrete cockpit controls. Pod outputs consist of RGB video to the colour head-down display and composite monochrome video to the head-up display. The pod is also fitted with an 8mm sealed video colour recorder, a data transfer unit and a video camera with a low light capability.

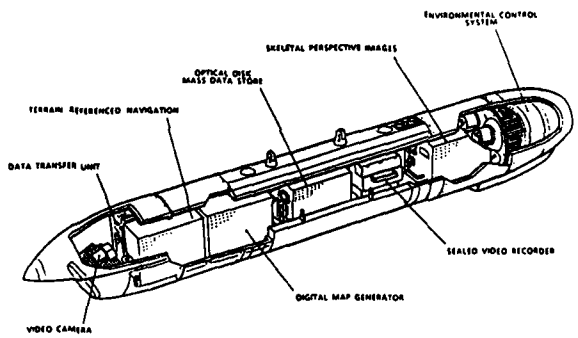



**FIGURE 2**
**COVERT MISSION SYSTEM DEMONSTRATOR**

The major modules of the demonstration system are housed in separate boxes to facilitate modifications required by the flight trials. The system can, however, easily be packaged into a single box for internal installation.

## NAVIGATION AND GUIDANCE

### Terrain Referenced Navigation

Precise navigation to an accuracy of a few tens of metres is essential if head-up perspective images are to overlay correctly the actual terrain. Precise navigation also reduces pilot workload as the moving map display faithfully indicates the exact position and shows what features can be expected. In the PENETRATE system this precise navigation is obtained from a terrain referenced navigation system.

Unlike a terrain following radar, the TRN system knows what the contours are like behind the next hill. This allows the pilot to follow the contours of the ground more closely than is possible with radar. Ballooning over ridges is avoided and the best advantage can be made of the available terrain screening. Automatic terrain following can also be implemented by coupling the TRN into the flight control system.

## Intelligent Ground Proximity Warning System

With an accurate knowledge of the surrounding terrain, together with aircraft present position, attitude and performance, the system provides sophisticated ground proximity warning. This "Intelligent" Ground Proximity Warning System (IGPWS) does not rely on historical radar altitude information and a flat earth. Instead the system continuously computes the g required to clear the data base terrain ahead. Two stages of warning are provided, which are determined by the g required and the performance margin available.

## DIGITAL MAP DISPLAYS

Low level flight is very demanding. The maximum time must be spent head-up monitoring the flight path of the aircraft in relation to the ground and attending to the overall tactical demands of the formation and the mission. The pilot must know where he is at all times and where he is going in relation to his planned waypoints, the local terrain, threat zones and his target. This information must be presented in a form which is easy and quick to assimilate. Superfluous information must be removed and important features such as masts and pylons must be highlighted.

Digital terrain elevation data is now available for large areas of the world and can be obtained by stereoscopy from military and commercial satellites. Where an existing digital data base is available, this allows the full capability of the digital map system to be used. Unfortunately, a full digital cultural database is unlikely to be available for some years and digitized paper charts will therefore be required initially to allow world wide coverage. Recognising the limited availability of digital data, the Ferranti digital map generator is configured to handle digital maps, digitized paper maps, or a combination of both.

## Overlay Capability

The digital map display shows navigation and intelligence information overlayed on the map, in a similar manner to the way pilots previously annotated their hand held maps. The PENETRATE system contains the normal map scales 1M, 500K and 250K, with 50K for IP to target runs. In addition, a 'route overview' 1:5 million scale map is provided to enable the complete route shape to be viewed on the screen. The intelligence overlay includes threats such as SAM sites, FLOT, FEBA and entry and exit gates. The information can also be colour coded to reflect category, importance or staleness. Masts, pylons and large vertical structures are available as a separate feature overlay. This enables important flight safety information to be highlighted.

The digital terrain elevation data base can be electronically overlaid on either digitized paper maps or true digital map features. This superposition adds a third dimension to the map display and offers several additional capabilities.

a. Sun Angle Shading

Sun angle shading of the terrain from any angle can be used to give a three dimensional effect.

b. Contour or "Safety Height" Shading

By selection, all terrain above the current aircraft height can be shaded, for example in red to highlight dangerous terrain during an instrument descent. Another alternative available is to colour terrain which is less than 1,000 ft below the current aircraft height.

c. Intervisibility Shading

To allow optimum terrain masking and minimise the overall effectiveness of any threat (such as a SAM site), the PENETRATE system computes threat zones and displays these as functions of aircraft height. The radial lines are displayed every 5 degrees and show those areas where line of sight intervisibility calculations indicate that the aircraft would be detected and vulnerable if it maintained its present height above the terrain. In the example shown in Figure 3, if the planned route through the hills to the top left is blocked by low cloud, the pilot could divert up the valley to the right without coming under threat from the SAM site on the ridge, provided he kept at the same height above ground. The extremities of the radial lines indicate the theoretical maximum range of the threat at that height, but this should obviously be treated with caution.

Intervisibility displays can also be used to indicate terrain which is hidden from the aircraft. This presentation can be used to allow semi-covert use of sensors such as radar, by indicating the areas from which their emissions are unlikely to be detected by ground based equipment.

## HEAD-UP DISPLAY ENHANCEMENT

High speed low level flight is demanding even in daytime and good visibility. At night and in poor weather outside visual cues and the FLIR image are degraded and the pilot's workload increases considerably. The PENETRATE system allows the pilot to enhance his forward view as the visibility decreases. The type of enhancement depends

on the weather conditions and the quality of the image available from the electro-optic sensors.

**FIGURE 3**
**INTERVISIBILITY DISPLAYS**

**Obstruction and Target Cueing**
In good conditions, only obstruction cues are required. Obstructions having significant vertical extent such as pylons, masts, chimneys and tall buildings are held in the obstruction data base. Obstruction symbols are then displayed in the correct perspective position in the HUD where they should overlay and therefore highlight the potential hazard, Figure 4. Hidden line elimination techniques are employed to remove any portion of the obstruction symbols which are obscured by intervening terrain. This elimination of hidden lines is very important, otherwise the obstruction will appear to be located in a false position much nearer to the aircraft. Clutter is avoided by reducing the luminance of obstruction symbols which do not present an immediate hazard.

**Ridge Line Displays**
In poor visibility and at night, the basic FLIR picture can be enhanced by the addition of ridge lines which are displayed exactly overlying the contours of the outside world, Figure 5. A ridge is defined and highlighted when the ground contours are tangential to the pilot's direct sightline.

**FIGURE 4**
**OBSTRUCTION CUES**

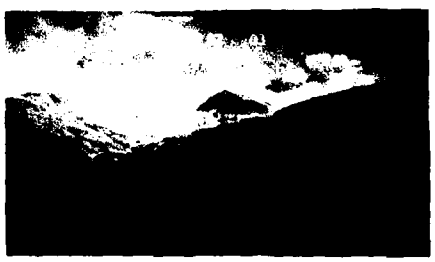


**FIGURE 5**
**RIDGELINE ENHANCEMENT**

**"Measles" Enhancement**
If every terrain elevation data point is shown in the HUD by a dot in its correct perspective position, the pattern of dots can enhance the ridgeline display by providing additional perspective information, Figure 6.

**Lattice Display**
In very poor conditions the elevation data points can be joined by lines to present a synthetic three-dimensional lattice, Figure 7. In the foreground, each lattice grid has 100 metre sides, which is the resolution of the basic digital terrain elevation data base. The grid resolution is widened with distance to present a uniform lattice density. The luminance of the lattice can also be varied with distance. All of these head-up skeletal displays are updated at 25 or 30 Hz (the video frame rate) and there are no limitations in aircraft speed or manoeuvre.

**Display Control**
Many combinations of perspective image enhancements are available as intermediate selections. One example is obstruction cues and ridgelines in the foreground with lattice in the background. In the PENETRATE system, a rotary "enhancement" knob is provided so that the pilot can select the type and degree of enhancement required, Figure 8. Inevitably, there is a compromise between clutter and enhancement. As the

visibility deteriorates, the pilot increases the display enhancement to the required level; when the visibility improves he turns down the enhancement to declutter the display.

FIGURE 6
MEASLES ENHANCEMENT

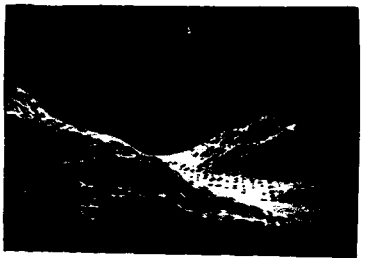


FIGURE 7
LATTICE ENHANCEMENT

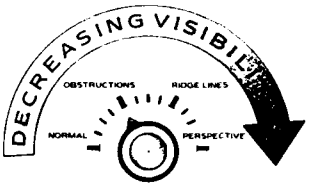



FIGURE 8
ROTARY ENHANCEMENT CONTROL

Due to the number of obstructions, ridge lines or lattice lines being displayed, the HUD can become congested or cluttered. Clutter obscures the FLIR image rather than enhancing it. To ensure that the FLIR picture still remains clearly visible, it is necessary to reduce the luminance of the overlay, except where an obstruction symbol is cueing an immediate hazard. One way of accomplishing this is to merge or add the raster overlay to the FLIR video and assign higher luminance levels to the flight symbology and immediate obstructions, whilst allowing less prominence to other enhancement symbols or overlays.

**HIGHWAY IN THE SKY (HITS)**

Most modern aircraft have a flight director symbol which shows the heading to steer to reach the next waypoint or target. If enough waypoints are inserted it is even possible to designate a complicated route of many segments. A two dimensional flight director, however, can only indicate the heading to steer at any one moment in time and requires constant attention if heading changes are not to be missed.

The PENETRATE Highway In The Sky (HITS) overcomes this problem by displaying the required 3-dimensional flight trajectory on the HUD superimposed on the terrain ahead in a form which is simple and natural to follow, Figure 9. The highway not only indicates the instantaneous heading required, but also shows the required flightpath several seconds ahead. The pilot is thus able to anticipate flightpath changes and has more time to devote to other activites.

The highway can be used both for route navigation and for recovery to a permanent runway or a tactical minimum operating strip. The highway symbols under initial evaluation are as a series of ground stabilised bars with upward pointing ends. The pilot flies along this highway without going below the bars.

**DATA BASE INTEGRITY**

The integrity of the data used by this covert mission system is subject to possible errors at three stages:

a. Digitization

b. Processing and Storage

c. Display Generation.

**FIGURE 9**
**HIGHWAY IN THE SKY**

## Digitization

The basic terrain data can be obtained from existing paper charts, photographic survey plates and satellite imagery. The Digital Land Mass Survey (DLMS) divides the terrain data into two basic categories, Digital Terrain Elevation Data (DTED) and Digital Feature Analysis Data (DFAD). The true digital vector or structured feature data base is presently only available for sample areas. As an interim measure, paper charts are, therefore, scanned to provide this cultural information.

### Elevation Data

The Digital Terrain Elevation Data (DTED) has a specified basic accuracy . Additional errors can be introduced during the digitization process and these errors depend on the digitisation technique used. If existing paper map contours are traced manually or automatically, inadvertant misinterpretation of contour lines can occur where lines are closely spaced or 'broken' by other cartographic information. If data is obtained by stereoscopy from military or commercial satellites, survey control points are needed to ensure accuracy; these points may be difficult to survey in unfriendly territory. Samples of DTED used by Ferranti have also been found to contain a few 'wild' vertical data points, though these can generally be detected and corrected during the data preparation procedure.

### Cultural Data

Errors in the cultural cartographic data base become numerous as the landscape evolves. Woods are cut down, new roads are built and old roads re-aligned. New buildings constantly change the shape of towns and villages. The accuracy and fidelity of a map is, therefore, proportional to the age of the source material. Fortunately, errors in the cultural data base, which have traditionally been a problem for human navigators, are not important to an inertial terrain or satellite referenced navigation system. Cultural modifications without significant vertical extent are also not critical to flight safety. Earthworks such as embankments, quarries and slag heaps are more serious, but only a few of these are of significant vertical extent and can, therefore, be categorized as obstructions. Small landscape perturbations of this nature are not significant to the accuracy of the terrain referenced navigator.

### Obstruction Data

Vertical obstructions such as masts, high buildings and electricity power lines are one category of cultural data which is particularly difficult to digitise, as it requires significant human intervention. As well as being wrongly recorded in both height and position, obstructions can be newly built, modified, demolished and even mobile (eg construction tower cranes and barrage balloons). An uncharted obstruction can be as lethal as an unknown surface-to-air missile (SAM) site. Obstruction data must, therefore, be treated in the same way as other military intelligence, as both have many similarities in terms of unpredictability, staleness of data and location errors.

## Processing and Storage

Data processing and compression may introduce errors; software designed for this purpose must therefore be treated as 'safety involved'. Careful processing can also detect errors. A simple check that the base height of each obstruction lies on the data base terrain is an obvious example. A continuity check of electricity line pylon positions can also highlight misplotted pylons. The optical disk mass storage medium has a basic error rate of about 1 in $10^5$. By applying error correction techniques these errors are reduced to less than 1 in $10^{13}$.

**Display Generation**

As well as the established error correction techniques, the system modules all have extensive built in test circuitry which ensures that the display is a faithful representation of the raw data.

**ADDITIONAL INTEGRITY**

In peacetime, additional integrity can be obtained from active sensors such as radar or lasers. In suitable conditions, a scanning laser can be used for ground and obstacle warning, but even very powerful lasers are attenuated by weather and cannot see more than about twice the range of the human eye. In very poor conditions, the radar may have to be used, albeit sparingly. The intervisibility display can then be used in reverse, to show suitable transmission periods when the radar is unlikely to be detected by known defences.

**THE PILOT AS A MONITOR**

The human brain is extremely good at pattern matching and the pilot is presented with a synthetic terrain picture which should exactly match the outside world. The TRN navigation error is generally a function of the roughness of the terrain. The degree of correlation and uncertainty is known within the system and this can be used to give the pilot warning of how well the system estimates it is achieving its terrain matching objectives. This 'navigation uncertainty' can be displayed on the head down map display and on the head up perspective display as 'metres error'. The synthetic terrain overlay may be in error from the real world, both laterally and in height. With the ridgeline, measles and lattice display enhancements, a mismatch in height (Z) could be misinterpreted as a longitudinal error (Y) and vice versa (Figure 10).

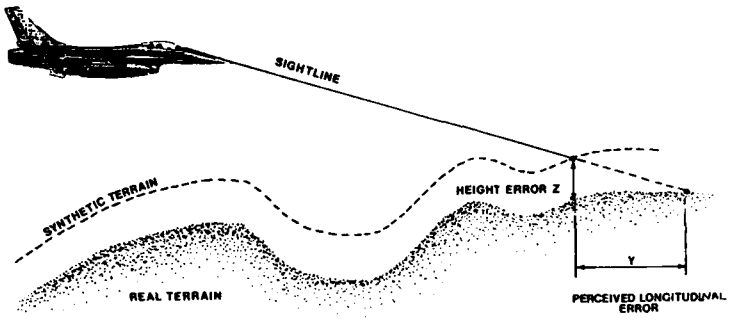



**FIGURE 10**
**HEIGHT ERROR INTERPRETED AS HORIZONTAL ERROR**

With the navigation accuracy currently being obtained from TRN systems, registration errors at long range between the perspective images and the real world are negligible. At very short ranges, however, the errors can sometimes become significant, particularly in the vertical plane.

In poor visibility and at night, if the perspective terrain image displayed in the foreground matches the real terrain as seen visually, or as seen by the FLIR or NVG sensors, then the pilot will have considerable confidence that the terrain image displayed in the background will accurately represent the ground that is not yet visible. When automatic Terrain Following (TF) is being used, the pilot can give his full attention to assessing the sensor and synthetic terrain information and to monitoring the way the autopilot is achiving the flightpath directed by the system. The ability of the human brain to assimilate dissimilar information and assess risk factor is unique. The ability to use this risk analysis to assess how the aircraft is progressing and to take action if necessary, can really only be a pilot function. The more complex the task the more attention is required.

Modern data storage methods give the aircraft system a considerable capability to fly safely and quickly without any tell-tale emissions, but the safety is only as good as the information in the data bank. The pilot remains the key.

**CONCLUSION**

Forward Looking Infra Red sensors enable a tremendous increase in operational capability by allowing the pilot to see ground features ahead of the aircraft in poor visibility and at night. They therefore extend the low level operational capability by a considerable amount. The PENETRATE system further enhances the pilots forward view and allows him to continue at low level with degraded visual or sensor displays. By displaying the terrain profile well beyond visual range and by cueing the approach of vertical obstructions, the PENETRATE system greatly enhances the safety of low level flight in both peace and war.

# TECHNIQUES FOR TRANSIENT ERROR RECOVERY AND AVOIDANCE IN REDUNDANT PROCESSING SYSTEMS

Stuart Adams
Mark Dzwonczyk

October 1989

*Fault-Tolerant Systems Division*
The Charles Stark Draper Laboratory, Inc.
555 Technology Square
Cambridge, MA 02139
USA

## SUMMARY

As the trend for increased storage in reliable, high-performance guidance, navigation, and control systems continues, coverage of transient memory failures becomes an increasingly critical problem. This paper discusses new techniques of recovery from such failures in redundant processing systems which perform high-frequency iterative control algorithms for flight critical GN&C. Two approaches are presented. The first employs hardware assisted recovery techniques to detect which memory segments in the failed processor need to be restored, so that recovery can be accomplished incrementally, by only restoring segments of memory which have been corrupted. The second approach is utilize a common fault-tolerant memory which allows errors to be masked and corrected on-the-fly eliminating the need for recovery.

## PROBLEM

The increased functional requirements of advanced guidance, navigation, and control electronics has necessitated the use of more powerful digital computer architectures. In addition to processing throughput, storage requirements are also increasing, as there is more data being collected and processed by larger, more complex software programs. The increases in chip density of semiconductor memory has lead to the illusion that there is little or no penalty to pay for increasing on-board storage. In fact, we have seen that semiconductor memory continues to be the leading contributor to digital system unreliability [1]. Systems for critical real-time control must account for the high failure rates of these components in order to preserve function integrity.

One approach to the design of highly reliable real-time systems is Triple Modular Redundancy, or TMR. Here, processing components form tightly-synchronized triplex systems. Hardware voters mask failures in real-time. Redundant processors run identical copies of program code allowing the operating system and voting/synchronization hardware to make the fault-tolerance aspects of the system transparent to the applications programmer. This approach has proven to be practical for achieving very high levels of reliability with only a small throughput penalty for fault tolerance.

An error in these systems will be detected and masked in real-time. A permanent fault will most likely cause the erroneous processor to consistently be in error. However, even most transient faults, such as temporary memory bit-flips, will often cause a processor to diverge from the majority computational stream. A continuous stream of voter errors will then ensue, not because the processor has any physical problem, but because it suffers from corrupted memory. A processor which exhibits persistent errors is taken off-line to facilitate degradation if further faults should occur. However, if the failure is transient, the reliability of the system is significantly increased if we are able to recover the processor rather than taking it off-line permanently.

A majority of the research and modelling of fault-tolerant systems has considered only permanent failures, but several studies have shown that the rate of occurrence of transient errors is 5 to 100 times that of fixed failures [2, 3, 4, 5]. Additionally, in nuclear or spaceborne environments one may expect transient failures even more frequently due to high radiation. One practical example is the recent loss of the Phobos 2 probe to that Martian moon. It is hypothesized that the loss was caused by a single event upset in the computer memory due to solar particles [6]. Our studies have shown that the ability of a triply redundant system to recover from such transient errors can decrease the probability of system loss by nearly an order of magnitude [7].

In the current CSDL Fault-Tolerant Processor (FTP) technology [8, 9] the process of bringing a redundant processing

channel back on-line after an error is termed *realignment*. The approach is to periodically attempt to re-synchronize the failed channel with the healthy ones. If the off-line member responds, real-time operation is suspended, THE off-line member's RAM is reloaded from the on-line members' RAM, and real-time functions are restarted. Current FTPs realign RAM at a rate of 1 Mbyte/sec while all other operations are halted. For typical cases, critical data in RAM occupies roughly 60K bytes (program storage is in ROM). The whole realignment procedure for this scenario takes approximately 120 ms, suspending the flight code for three 40 ms iterations during recovery. The delays associated with recovery are greater when larger RAM segments in the faulty channel must be restored while maintaining full functionality of control algorithms with higher iteration rates. Clearly, better techniques are needed for transient error recovery and avoidance in current and future systems which require more memory and higher performance.

We have investigated two approaches to this problem and present them here. First, transient error recovery is discussed using a novel method of Segment Access Signatures. The latter portion of the paper suggests architectural methods for avoiding (masking) momentary failures in the storage subsystems.

## ERROR RECOVERY

We first introduce a new method for error recovery (channel realignment) denoted Segment Access Signatures (SAS). This technique uses some monitoring hardware which connects to a processor's address and data busses. The SAS hardware contains a signature memory of $M$ words. The main memory store of $N$ words is arbitrarily divided into $M$ segments of size $N/M$ words with each signature word in the SAS memory having a one-to-one mapping to a particular segment in main memory (Figure 1). The signature word for each segment represents all bus accesses to that segment. A bus access consists of a CPU read or write cycle with an address and data being presented on the bus. For each bus access monitored by the SAS hardware, the access signature word for the particular segment is updated by computing a check code, such as a checksum or CRC, of the current value of the access signature word, the address of the word accessed within the segment and the data value presented on the bus. Note, that this access signature value is not representative of all values in the segment, but only accessed values and their addresses. The signature is a unique value representation of the sequence of accesses to the memory segment and is dependent upon both the data and addresses read/written to the segment.

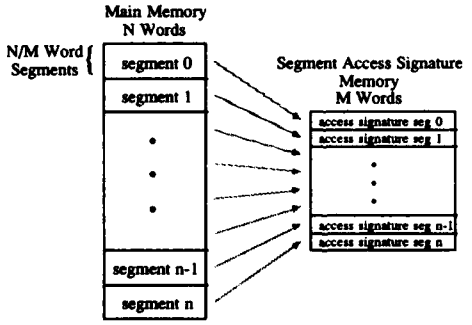



***Figure 1.*** *Mapping of segments to access signatures.*

Read, write or read/write access signatures can be computed. If signatures are performed on read accesses, then the signature will represent the computational flow of the processor since memory reads are performed for both instruction and data fetches. However, if only write access signatures are performed, then the signature for a particular segment of memory represents a sequence of modifications by the processor to that particular segment of memory. Given a known initial condition of the memory segment and a known initial value of the access signature for that segment, the write access signatures represents the state of the segment, since changes from the initial condition can occur only by the processor writing to the segment (except for latent memory bit flips internal to the RAM which will be discussed shortly).

If we compare write access signatures at two points in time for the memory of a single processor, we can identify segments within the memory which have changed over that time period. We can also use write access signatures to discern differences in the internal state between redundant processors, assuming the processors started with identical initial conditions.

The algorithm used to compute the access signature may be any algorithm which is suitable for computing a signature

or check value on a continuous stream of data. Both checksum and CRC checks are viable options. The number of bits in the signature and the spectral properties of the signature algorithm will determine the probability of the same signature occurring for two different access sequences. For example, if we assume a spectrally independent 32-bit signature algorithm, then the probability of the same signature occurring for two different access sequences would be $2^{-32}$.

Figure 2 depicts the architecture of the Segment Access Signature hardware for a processor with 1 Mbyte of memory. For this example, the SAS hardware has 2K 32-bit access signature words and is using two 32-bit CRC checks to compute the access signatures. During a memory access, 32 bits of data and 20 bits of address are presented on the system bus. The upper 11 bits of the address are used by the SAS hardware to select one of the 2K access signatures corresponding to the segment of memory being accessed. (Since there are 2K SAS words, each segment contains 512 bytes.) The old segment access signature is read from the SAS memory and a CRC is computed using this value and the 32-bit data value on the bus. The result of this CRC is fed into another CRC computation combined with the lower 9 bits on the address bus. The resulting CRC – which was computed from the original signature, the data being read/written on the data bus, and the address within the segment – is now stored back into the SAS memory over the signature word.

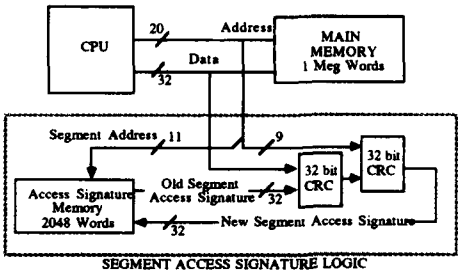



***Figure 2.*** *Segment Access Signature Hardware*

The SAS hardware is a completely passive monitoring function that can be readily layered upon existing hardware/bus architectures. The only throughput requirement of the SAS hardware is that it be able to perform the signature updates at a sustained rate equal to the maximum access rate of the main memory. With the use of pipelining for signature computation, the bottleneck is simply the necessity to perform a SAS memory read and write for each access to main memory. The SAS hardware would typically be implemented on a single semi-custom chip which would have on-chip memory accesses times of less than 5 nanoseconds, making implementation feasible for very fast main memories.

When using access signatures to identify the sections of memory which have changed or are corrupt, the processor must compare its current signatures with those of another processor or those of the same process from a previous time. For example, to identify a single corrupt 512 word segment in the example in Figure 2, we must compare the 2K 32-bit signatures. To reduce the time consuming process of exchanging or saving this information between processors or sample times and performing the comparison of 2K words to identify a single segment, we can use a hierarchy of signatures as depicted in Figure 3. The processor would begin by comparing master access signatures to determine if any segments were different. Then the level 1 signatures would be compared, and only those level 1 signatures which miscompared would need further examination.

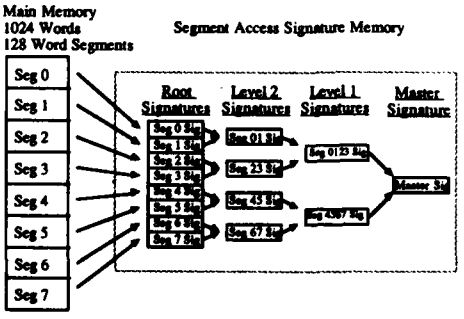



***Figure 3. Mapping of Hierarchical Signatures***

We next discuss how segment access signatures can be used for recovery in redundant processing systems in two different ways. The One Shot Recovery scheme utilizes access signatures to discern differences in internal state between redundant processors. Running Recovery utilizes access signatures between iterations to determine segments within the processor which have changed during the iteration.

**One Shot Recovery**

*This section discusses the use of Segment Access Signatures for what we term One Shot Recovery. One Shot* Recovery is when recovery of a failed processor is accomplished in one operation by using SAS to detect which memory segments in the failed processor need to be restored, such that recovery can be accomplished incrementally, by only restoring those segments of memory which have been corrupted.

If we assume that initially, memory segments and segment access signatures in all redundant processors are identical, then under non-faulty conditions the access signatures of all redundant processors should remain identical during normal operation, since the processors are running identical code synchronously. If a transient fault occurs and causes a processor to be taken off-line, the processor's memory may be affected in one or more of following ways:

I. A bit-flip in one or more memory cells occurs or data is written incorrectly internally to the RAM. This type of error is latent until the processor performs a read from this memory location. This fault would be the direct cause of a transient and not simply a side effect of the error.

II. The processor places bad data/addresses on the bus, or the data/addresses are corrupted by noise on the bus when performing a write operation. This fault would also be the direct cause of a transient and not simply a side effect of the error.

III. The processor writes valid data to valid address, which is logically incorrect (from the majority viewpoint). This type of error is not a direct result of the transient but rather a result of the processor's subsequent actions after *acting upon a faulty piece of data from an external sub-system, an error internal to the CPU, or one of the above* two faults. Neither the memory, nor the processor are faulty. Rather they are operating on faulty input causing the computational stream to diverge from the majority.

If only write access signatures are used, the access signatures in the faulty processor will have signatures which vary from the majority's signatures for those segments which have been corrupted by errors of type II and III, but will not detect type I errors. If read access signatures are performed it will be possible to detect type I errors. The faulty processor will have read a data value different from that read by the non-faulty processors causing the faulty processor to have a different signature for that segment.

By providing access signatures which are updated by both reads and writes we can accomplish the detection of all types errors which would corrupt a faulty processor's memory (types I, II and III). Recovery can then be accomplished by restoring only those segments which have been corrupted as designated by non-matching signatures. However, by updating signatures on read accesses, we will update the signatures for accesses by the faulty processor of instruction and data fetches for a program which may be out of control due to a faulty input. *This would cause segments to be marked as corrupt if* the faulty processor performed a read of a non-corrupt instruction or data value which was not performed by the majority. The thousands of instruction and data fetches that a processor routinely performs would most likely cause an excessive number of *segments to erroneously be marked as corrupt should a faulty processor follow a computational path other than that of the* majority.

Since the major contributor to the high incidence of transient errors is from bit errors internal to the RAM, it is not sufficient to just perform signature updates on write accesses. An alternate method of providing coverage of type I errors without performing signature updates on every read access is to use a traditional parity error detection scheme. Read access signature updates would then occur only on read accesses which had a parity error, causing the access signature for the segment with the parity error to differ from the majority's signature for that segment.

One Shot Recovery is dependent on the assumption that a transient fault will only corrupt a portion of a processor's memory which is small enough to be recovered in one operation without degrading critical real-time functions. Although this may be true for most transients, the effects of a transient may completely scramble the processor's memory such that no assumptions about the contents of the memory may be made. To address the issue of this type of transient we must turn to a *more robust recovery method called Running Recovery*.

**Running Recovery**

This section introduces the concept of Running Recovery. Running Recovery is different from One Shot Recovery in

that it makes no assumptions about the state of the failed processor's memory, and recovery does not occur in one operation but occurs over a number of iterations. Running Recovery is necessary when recovery cannot be completed within one iteration. This may occur for the following reasons:

1. The access signatures indicate that the amount of data to be restored is too great to be done in one operation while maintaining full functionality of control algorithms.
2. One Shot Recovery has been attempted and failed several times. This may occur if an access signature erroneously indicates that a segment has not been corrupted thus preventing it from being restored.
3. A complete main memory, or SAS memory loss has occurred due to power loss, off-line repair, a run away program, or suspected SAS memory corruption.

In all of these cases we must assume that the processor has to be re-aligned from scratch. We now use segment write access signatures to determine which segments have been altered from one iteration to another in the non-faulty members. The faulty processor being aligned runs no code but complies with align instructions from the non-faulty processors and awaits a start signal. A small portion of each iteration of the operational code is designated for performing recovery until recovery is complete. Each iteration, the non-faulty processors determine which segments in their own memory have changed since the last iteration. They then restore these segments into the faulty processor. In the remaining time, if any, the non-faulty processors also restore as many other segments as time allows. Assuming that the restoration of segments which have changed during that last iteration does not occupy the full duration of the allotted re-alignment interval, there will be an opportunity to restore some non-changed segments with each iteration. Over a series of iterations the number of non-restored segments will dwindle until at some point all segments will have been restored and the faulty processor can be recovered.

The algorithm in Figure 4 is used by the non-faulty processors to restore the faulty processor. The faulty processor simply restores segments upon command from the majority until it receives a signal that restoration is complete. The routine *restore_seg(y)* restores segment *y* in the faulty processor's memory from the non-faulty members' memory. Successive calls to the function *get_next_changed_seg* return the numbers of segments which have changed since the last call to the subroutine *save_signatures*. The function returns -1 when no changed signatures are left. We assume that initially $x$=0, and *restored[0...N-1]*=0 where *N* is the total number of segments. The algorithm is called once per iteration until *restored[0...N-1]*=1. The variable *max_seg_restores_per_iteration* indicates the number of segments that can be restored in the time slot allocated for recovery in each iteration.

```
Once_Per_Iteration do
     begin
     segs_restored=0;
     while segs_restored < max_seg_restores_per_iteration do
          begin
               y=get_next_changed_seg();
               if (y ≠ -1) then do
                    begin
                         restore_seg(y);
                         restored[y]=1;
                         segs_restored=segs_restored+1;
                    end
               else
                    if (restored[x] = 0) then do
                         begin
                              restore_seg(x);
                              restored[x]=1;
                              x=(x+1) mod N;
                              segs_restored=segs_restored+1;
                         end
               endif
          y=get_next_changed_seg();
          while (y ≠ -1) do
               begin
                    restored[y]=0;
                    y=get_next_changed_seg();
               end
          save_signatures();
end
```

***Figure 4. Processor Recovery Algorithm***

Running Recovery assumes that if recovery is to occur that there exist a series of *S* iterations over which no more than (*S* x *max_seg_restores_per_iteration* - *N*) segments will have changed in the non-failed elements. Recovery can still occur

even though periods in which the number of changed segments, since the last iteration, is greater than the maximum number of segment restores possible per iteration.

The algorithm of Figure 4 readily illustrates how recovery can be accomplished even when as much as *(max_seg_restores_per_iteration* - 1) of the alignment interval is spent aligning segments which have changed since the last iteration. However, ideally one would like to align the segments changed least frequently first: Repeatedly aligning a segment, which always changes, each iteration is wasted time which would be better used aligning a segment that probably would not need to be aligned again. To perform a weighted selection of segments, there would need to be a "changed" count for each segment to allow for the selection of least changed segments first. The overhead required for maintaining this count would need to be implemented in hardware since the updating and selection of least weighted segments could consume considerable overhead.

The Running Recovery scheme uses segment write access signatures to simply detect which segments have been modified by a write access. The signature information insuring that the correct information was written at the correct address is not really needed. Thus, we need not implement a full 32-bit signaturing scheme for each segment but rather may use a much simpler 1-bit flag which is set when a write access occurs in a segment. To accomplish Running Recovery with this simpler Segment Write Marking approach, all segment write flags are initially set to "0" at the beginning of an iteration. Upon completion of the iteration, access flags which are set to "1" indicate segments which have been changed. The use of a hierarchy of access flags, similar to the hierarchy of access signatures shown in Figure 3, is still needed to readily identify segments which have changed.

Favorable experimental results using the SAS techniques have been seen and presented elsewhere [7]. For the sake of brevity, we omit them from this discussion.

## ERROR AVOIDANCE

A second approach to tolerating transient faults in critical systems is by masking the failures before they can corrupt a processing subsystem. To complement the work in error recovery described above, research in real-time fault-tolerance via coded redundancy is also currently being pursued. In this approach, transient faults are corrected as they occur using conventional error correction mechanisms.

As discussed above, TMR systems typically obtain their ultra-reliability through full replication of all hardware and software elements. This includes processors, communication and input/output ports and storage elements. The replication of high-speed random-access memory (RAM), however, has several disadvantages for reliable systems in a practical setting. The first, noted previously, is, due to the high susceptibility of semiconductor devices to failure, RAM is the principle contributor to system unreliability in digital computing systems [1]. A second, sometimes more pragmatic, disadvantage of semiconductor RAM is its high cost. Although price per bit continues to decline as density increases, the increased requirements for storage in digital control systems have outpaced the drop in cost. For example, the 256 x 4 (1K bit) memory chips used in the European Space Agency's Ulysses and NASA's Galileo spacecrafts cost roughly $20,000 each in 1988 (about $1M for 6K bytes) [10]. The Imaging Orbiter for NASA's slated Mars Rover Sample Return (MRSR) mission is expected to require 50 Mbytes of on-line memory [11]. If one projects RAM fabrication technology to avail 1 Mbit chips by the MRSR technology freeze, and assuming a comparable price per chip, the 500-chip memory would require a $10M outlay.

However, triplex redundancy is not always necessary for memory subsystems. In fact, such storage systems can be designed to tolerate internal transient failures [12, 13]. In these systems, coded redundancy is utilized to prevent single or multiple bit errors from corrupting an entire data word. This technique can be utilized in the FTP architecture with a global memory system [14, 15], i.e., one encoded common memory which is available to all replicated processors. Coding techniques can allow for failures of portions of memory while sustaining the integrity of the encoded data, thereby providing high-reliability with a much smaller amount of additional storage (approximately 125% v. 300%).

The fact that the memory subsystem can tolerate transient failures obviates the need for recovery. For advanced space applications such as the MRSR mission, where storage requirements will be extremely large, more durable and cheaper storage mediums can be employed for the global memory. Conventional caching techniques can then be used to compensate for performance penalties which may be incurred due to the access times of more robust media.

The remainder of this paper discusses issues of a storage architecture currently under investigation: a single extremely reliable memory shared by all replicated processors. Reliability for this memory is increased using a coding scheme which can

provide detection and correction of up to some number of bit errors per word. Access times to the coded global memory need not be extremely fast if the memory is cached to the FTP core through a high speed buffer.

**Global Coded Memories**

The use of a global coded memory for the FTP was first suggested by Davis [15]. In that design (Figure 5) words in the large global memory are divided into smaller symbols, allowing the data to be encoded to detect and correct limited numbers of bit errors in each word. The global memory is interfaced to each processor via a decoding channel (for read operations) and from each processor via an encoding channel (for write operations). The outputs of the encoding channels are passed through a voter to cover for FTP core failures. Since the FTP is instruction synchronized, operations to and from the global memory will be synchronized.

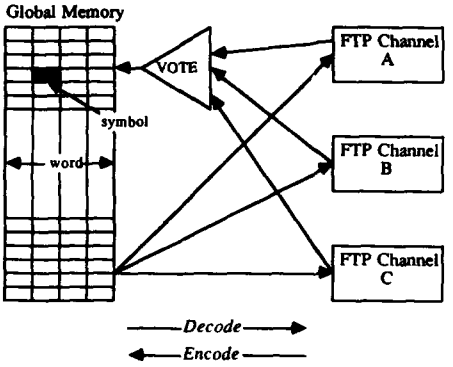



***Figure 5.*** *A Global Encoded Memory*

Coding schemes are available which can store (encode) and retrieve (decode) data from the memory in the presence of any symbol failure. The basic premise of coding is to divide a stored element of $m$ bits into $M$ equal size pieces, or symbols, and add $P$ parity symbols to obtain $(M + P)$ total symbols. The $P$ symbols encode the M data symbols in such a way that the original $(M + P)$ symbols can be reconstructed from a garbled set of original $(M + P)$ symbols. Of course, the number of symbols which can be garbled at one time will greatly affect the code and the number of required parity symbols. For the sake of brevity, we refer the reader to any one of the excellent texts on the subject, one of which is [16].

Symbols within a data element ($m$ bits wide) must be electrically isolated in order to ensure the integrity of the code. This means that a properly implemented encoded memory system must not allow failures from one symbol corrupt any other symbol in the same element. If this were the case, a single failure could destroy a data element and thwart the encoding process.

A global encoded memory can obviate the need for realignment. Depending upon the robustness of the code, any number of failures in the memory can be tolerated. For example, if one symbol is cleared to all zeros, the data element can still be recovered intact. This applies to permanent as well as transient failures. When a corrupted word is rewritten to the memory at some later time, if the failure was momentary, the storage integrity will be preserved and the correct data will be rewritten. If the failure is permanent, that symbol will be permanently in error. If more failures are to be tolerated, appropriate number of parity symbols must be present along with a robust coding scheme.

**Reliable, Inexpensive Media**

The method presented above offers an alternative redundancy technique for reliable storage which can obviate the need for realignment and tolerate transient memory of data. It can also reduce the component costs to nearly a third of TMR methods. (A roughly 25% increase will be required in hardware to implement a robust code; 200% increase is needed for TMR.) However, for large memories, such as required by the MRSR mission, the cost savings will not be substantial enough.

Additional considerable savings can be gained if the storage medium is something other than semiconductor integrated circuitry. Although relatively cheaply available for benign environments, qualification of semiconducting devices, such as CMOS, for flight applications makes the technology a quite expensive media for mass on-line storage. For the global memory,

bubble, core or even tape cartridge memories may suffice as low-cost storage. These media offer high reliability at a lower cost than properly qualified semiconducting materials.

A main drawback with this cost-saving solution is that these media are much slower than the semiconducting counterparts. Thus, when the storage cells are replaced with a slower, more reliable medium, a severe performance degradation is expected. The solution of this problem is the topic of the next section: utilize high-speed caches to improve performance.

**Caching for Performance Improvement**

A major improvement to the design would be to utilize a slower but more reliable medium for storage without incurring a serious performance penalty. This can be accomplished by *caching* a very small subset (32K bytes, for example) of the global memory at each processor interface (Figure 6). The cache memory would be extremely fast and contain decoded data, lending itself to direct processor interface without an access time penalty. This method allows the FTP rapid access to data which is often used, while at the same time naturally migrates least used data back to the more reliable encoded memory. The architecture of this new design is shown below. In keeping with the requirement of data congruency for proper FTP operation, all local caches are identical.

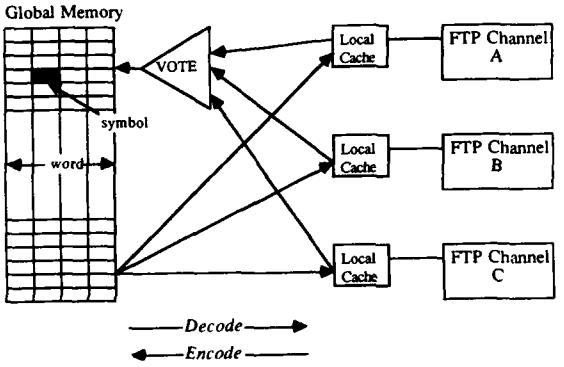



***Figure 6.*** *A Cached Global Encoded Memory*

The cache would operate much like high-speed storage in virtual computing architecture. A block of data from mass storage (the slow, reliable encoded memory) would be loaded one at a time. This is similar to swapping in a page of program memory in a the virtual computer architecture. Normal operation is then executed from this cache. When a page-fault occurs, that is, when the FTP needs access to an address not loaded in the cache, the current cache contents will be written to mass memory (voted and encoded to prevent errors from propagating into the mass storage) and the new block will be loaded into the cache. Hardware memory management units will be employed to facilitate page swaps.

This method of operation is commensurate with real-time operation since under most circumstances, mass memory will be used only for data storage. Program instructions will normally reside in ROM and replicated in each channel. With mass memory containing only data, the global memory will be organized such that related data is stored in contiguous memory locations. Related data will likely undergo execution at the same time. Thus page swaps will roughly follow task swaps as new sets of data is operated upon.

**Example of Use**

A good example of this are terrain maps to be used by an autonomous vehicle. In a typical scenario, the planning function of the control computer requires information only about its immediate area. This local vicinity information would be automatically migrated to the local cache. As the vehicle progresses in its travels, new portions of the map are loaded into the cache as the least used ones are removed. The continuity of map information in the address space of the global memory lends itself well to cache implementation. Figure 7 illustrates the idea.

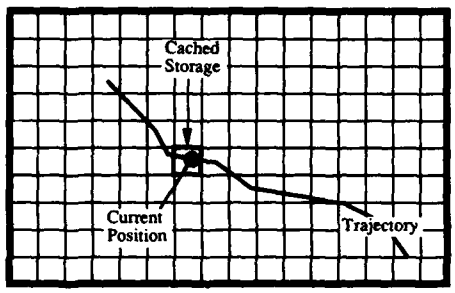



***Figure 7.*** *Map in Global Memory*

**Memory Failures**

Memory failures must now be discussed in two categories: global memory failures and cache failures. The data in global memory can tolerate a certain number of bit or symbol failures, depending upon the coding scheme. Although the coding mechanism will cover for localized bit errors, including transients and single event upsets due to environmental radiation, it cannot cover for a memory system which simply wears out. Thus, a reliable media is also required.

Cache data is error-prone (due to hardware failures) while it is in the local caches. A failure in the cache may corrupt an entire channel, but the voting and encoding ensure that the fault is contained within that channel. The failure cannot propagate to other channels or to the mass memory. Once the channel is brought back on-line, through the realignment process, the local cache is reloaded. This memory refresh process will resemble any other page swap and take an insignificant amount of time compared to a complete memory reload currently performed during channel realignment.

The cache, thus, resembles conventional high-speed storage in the FTP. The global memory allows reliable mass storage, accessible through the cache itself. Cache failures are handled like conventional memory failures in the FTP, but do not suspend real-time operation for any critical time because the size is small and loading is from the global memory. The global memory provides a compact method for reliable storage of mass data.

**REFERENCES**

1. M. J. Dzwonczyk, M. F. McKinney, S. J. Adams, and R. J. Gauthier, *Avionics Architecture Studies for the Entry Research Vehicle*, NASA-CR-181828, May 1989.

2. D.P. Siewiorek, V. Kini, H. Mashburn, S.R. McConnel, and M. Tsao, "A Case Study of C.mmp, Cm*, and C.vmp: Part I -- Experiences with Fault Tolerance in Multiprocessor Systems", *Proceedings of the IEEE*, Vol. 66, No. 10, October 1978, pages 1178-1199.

3. J.H. Wensley, "Effect of Transient Errors on Computer Control Systems", *Proceedings of the Third Annual Control Engineering Conference*, Rosemont, IL., May 1984.

4. W.K. Mikhail, R.W. Bartoldus, and R.W. Rutledge, "The Reliability of Memory Subject with Single Error Correction", *IEEE Transactions on Computers*, Vol. 21, (December 1972), pp. 1322-1331.

5. S. McConnel, D. Siewiorek, and M. Tsao, "The Measurement and Analysis of Transient Errors in Digital Computer Systems", *Digest of Papers, The Ninth International Symposium on Fault-Tolerant Computing (FTCS-9)*, Madison WI, June 1979.

6. M. A. Dornheim, "Latest Soviet Planetary Mission Plans Reflect Shift to Conservative Outlook", *Aviation Week and Space Technology*, August 28, 1989, pp. 21-22.

7. S. J. Adams, "Hardware Assisted Recovery from Transient Errors in Redundant Processing Systems", *Digest of Papers, The Nineteenth International Symposium on Fault-Tolerant Computing (FTCS-19)*, June 1989, pp. 512-519.

8. J.H. Lala, "An Advanced Information Processing System", *Proceedings of the IEEE/AIAA 6th AIAA-IEEE Digital Avionics Systems Conference*, December 1984, pp. 199-210.

9. J.H. Lala, L.S. Alger, R.J. Gauthier, and M.J. Dzwonczyk,"A Fault-Tolerant Processor To Meet Rigorous Failure Requirements", *Proceedings of the AIAA/IEEE Seventh Digital Avionics System Conference*, October, 1986, pp. 555-562.

10. Lenorovitz, J.M., "ESA Replaces Ulysses' Memory Chips, Computer During Launch Standdown", *Aviation Week and Space Technology*, April 18, 1988, p. 18.

11. Randolph, J.E. (ed), "Mars Rover 1996 Mission Concept (Partial Results of the 1986 Preliminary Study of a Mars Rover/Sample Return Mission)", *NASA Jet Propulsion Laboratory technical report JPL-D-3922*, 22 December 1986.

12. J.J. Stiffler, "Coding for Random-Access Memories", *IEEE Transactions on Computers*, Vol. 27 (June 1978), pp. 526-531.

13. W.K.S. Walker, C.W. Sundberg, and C.J. Black, "A Reliable Spaceborne Memory with a Single Error and Erasure Correction Scheme", *IEEE Transactions on Computers*, Vol. 28, No. 7 (July 1979), pp. 493-500.

14. T. Krol, "The '(4-2)-Concept' Fault Tolerant Computer", *Digest of Papers, The Twelfth International Symposium on Fault-Tolerant Computing (FTCS-12)*, 1982.

15. L.D. Davis, *Error-Correcting Memory for a Triplex Fault-Tolerant Processor*, MIT Master of Science Thesis, February 1984.

16. Gallagher, R.G., Information Theory and Reliable Communication, John Wiley and Sons, Inc. 1968.

# THE ROLE OF TIME-LIMITED DISPATCH OPERATION IN FAULT-TOLERANT FLIGHT CRITICAL CONTROL SYSTEMS

Deborah F. Allinger
Frank J. Leong
Philip S. Babcock IV
The Charles Stark Draper Laboratory Inc.
555 Technology Square
Cambridge, Massachusetts, 02139, U.S.A.

Richard F. LaPrad
Gary C. Horan
Pratt and Whitney
400 Main Street
East Hartford, Connecticut, 06108, U.S.A.

## ABSTRACT

The use of fault-tolerant system design concepts to achieve otherwise unattainable levels of reliability in modern flight critical control systems is rapidly becoming commonplace. Fault-tolerant flight and propulsion control systems, for example, are now being deployed in modern aircraft, spacecraft, and submersibles. The unique characteristics of these systems pose new problems for the designers of such systems and afford new opportunities for their users. A basic motivation for introducing fault tolerance is to be able to preserve some level of functionality of the system in the wake of failures of some of the system's components. This property of fault-tolerant systems affords an opportunity to dispatch these systems with failed components for a limited time period. This mode of operation is referred to as *time-limited dispatch*. In time-limited dispatch operation, benefits related to both maintenance and operations can be realized. Aircraft maintenance actions can be deferred until a more convenient time or place, for example. Similarly, the sortie rates that can be realized in tactical situations can be increased. In order to determine optimal or near optimal dispatch policies for fault-tolerant systems, one must have a systematic means of establishing dispatch policies and be able to quantify the benefits that can be realized by adopting specific dispatch policies. A tractable methodology for doing so is described and illustrated in this paper.

## 1. INTRODUCTION.

The rapid development of digital electronics and the requirement for improved performance have led to the introduction of digital control systems for modern aircraft, spacecraft, and submersibles. A critical design requirement for these systems is high reliability. This means that the control system must function reliably enough during the course of a mission to render the probability of a control system failure extremely small.[1] This high reliability is achieved by both the incorporation of components with high reliability and the careful management of redundancy available in the control system. It is precisely these two features that lead to the consideration of operations that incorporate aspects of time-limited dispatch.

Time-limited dispatch, in a broad sense, is a mode of operation that permits the use of the system for limited time periods even though there is knowledge that certain components in the system are not operational. The advantage of such an approach is that some maintenance operations can be deferred until the vehicle arrives at a more convenient time or place. This may lead to significant improvement in system performance by consolidating both the logistics and the expertise of maintenance operations.

To rigorously analyze and quantify the effects of time-limited dispatch operation on a system's performance and reliability, a new evaluation scheme has been developed by the System Evaluation and Operational Analysis Section of the Charles Stark Draper Laboratory, Inc. working in conjunction with the Pratt & Whitney Division of United Technologies, Inc. The basis for this scheme is a new technique known as time-limited dispatch reliability. Time-limited dispatch reliability is an analysis which uses various modeling methods for both the design of a time-limited dispatch mode of operation and also the evaluation of the subsequent impact on system performance. The efficacy of this analysis and its feasibility have already been demonstrated with respect to the PW4000 electronic engine control system [1].

In this paper a dual-redundant control actuation system that incorporates the salient features of both flight and propulsion control systems is used to illustrate the analytic techniques which permit the quantification of vulnerability to system failure in specific failure configurations. This example system incorporates redundancy management via fault detection, isolation and reconfiguration schemes. The performance of those schemes is reflected in the associated coverage parameters. The analysis presented permits the dispatch classification of each system component, and furthermore, the time limits for the time-limited dispatch are determined. Finally, techniques to quantify the impact on system performance given a time-limited dispatch mode of operation are illustrated.

---

[1] We will use the term *system failure* (SF) to mean that the control system has degraded to an unacceptable level during operation even though the system itself may not be completely failed.

## 2. OVERVIEW OF THE ANALYTIC OBJECTIVES.

Time-limited dispatch can be viable only if it does not adversely impact the reliability of the vehicle operation. This aspect can be viewed in two stages. First, given that a control system has a specific failure, what is the additional vulnerability of the vehicle to system failure (SF)? Second, what is the impact that time-limited dispatch operation will have on the control system's reliability and behavior. These two questions give rise to the two analytic objectives of time-limited dispatch reliability.

The first objective is the *specific configuration objective*. It is concerned with how specific system configurations can be classified according to their dispatchability. With respect to this objective, there are certain components whose loss can increase the vulnerability of system failure significantly enough that the potential risk may be too great to allow dispatch with these components failed. On the other hand, there are other components that the system operation does not so heavily depend on. Thus, the system may be able to to be dispatched for long times with these components failed. Finally, there is a group of components whose failures have a moderate impact on the system vulnerability. The nature of this impact is that the vulnerability accumulates as time goes on, eventually reaching an intolerable level. These are the components that can be time-limited dispatched. Therefore, an analysis of the impact on the vulnerability to system failure while operating the system with each component independently failed provides a means of classifying each component failure into a category according to its dispatchability. In a similar manner system configurations initiated by two or more faults can also be dispatch classified. From this information, a time-limited dispatch mode of operation can be developed in which various components or component configurations can have different repair or replacement schedules as determined by each one's dispatch status. Thus, the specific configuration objective has as its goal the *design* of a reliable time-limited dispatch mode of operation. Accomplishing this objective requires an analytical method which can easily evaluate conditional events. Our recent results [1] have shown Markov analysis to be an excellent choice for systems like the one described in the example below. These methods are illustrated in the remainder of this paper.

The second analytic objective is the *system impact objective*. It examines the impact that time-limited dispatch operation has on the system's performance. The performance of a control system is measured through a variety of figures of merit such as probability of loss of control, mean time between maintenance actions, mean time between unscheduled removals, and sortie rates. While each figure of merit reflects an operational goal, it is generally not possible to achieve or improve all goals simultaneously. For example, a mode of operation which significantly increases the mean time between maintenance actions will not, in general, also substantially reduce the probability of loss of control. Instead, operational goals involving issues such as safety and maintenance actions must be balanced against each other to achieve the best possible operational mode for a given set of constraints. Thus, in the broadest sense, the system impact objective is accomplished by solving for the optimal or near-optimal time-limited dispatch criteria with respect to a given set of constraints. In the example below, this objective is illustrated with respect to a single constraint which is called the *fleet-average target SF rate*.

## 3. A CONTROL ACTUATION SYSTEM: AN ILLUSTRATIVE EXAMPLE.

### 3.1. The Specific Configuration Objective.

In this section we give an illustration of how the specific configuration objective is analyzed with respect to a dual-redundant control actuation system that incorporates the salient features of both flight and propulsion control systems. Regardless of the specific architecture being investigated, the first step is to develop an analytic model of the system which can provide measures of rates at which fault conditions occur and the probabilities of various operational configurations. The model must reflect the system's status as component faults occur. Thus, we begin with a system description.

#### 3.1.1 The System Model

The system block diagram of Figure 1 depicts the organization of the components in our example control system. The two channels, A and B, are connected through the crosslink (XLINK) and are identical. Each contains two types of sensor elements (S1, S2), a central processing unit (CPU), and an actuator (ACT). The components' failure rates and coverage values are given in Table 1. The coverage value represents the probability of correctly detecting and isolating the failure of a component, given that a failure has occurred.

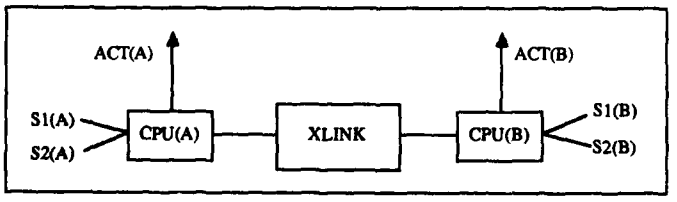



**Figure 1. System Block Diagram**

Table 1. Input Parameters

| COMPONENT | FAILURE RATE (1/hr) | COVERAGE |
|---|---|---|
| S1 | 3.00E-5 | 0.85 |
| S2 | 1.00E-5 | 0.95 |
| ACT | 5.00E-6 | 0.98 |
| CPU | 5.00E-5 | 0.95 |
| XLINK | 1.00E-6 | 0.98 |

When starting from a full-up or no-fault condition, the system initiates control in Channel A. The reconfiguration strategy employed attempts to use the sensors nearest the CPU that is in control. But if a given sensor's information is detected as faulty, it is replaced by the other channel's sensor data via the crosslink. Actuators, on the other hand, can be controlled only by their respective CPUs. Thus, if an actuator failure is detected, the system switches control from one CPU to the other in order to utilize the operational actuator.

The system has two operational control modes: P-mode (primary) and S-mode (secondary). The P-mode requires that at least one CPU and its associated actuator be operational and one S1 and S2 sensor be operational and accessible from the active CPU. The S-mode is identical to the P-mode except that only operation and access of the S2 sensor is required. The system switches control from one CPU to another if a better operating mode can be achieved. System configurations that do not achieve either P-mode or S-mode requirements are considered system failures.

From a description of the components in the system and its control modes, a Markov model is constructed by examining the ways in which components fail and the consequences on system performance. This model is represented as a graphic network in Figure 2, and the description of its states (nodes) is listed in Table 2. Notice that the states are grouped according to the number of failures which the system has experienced. Transitions between states occur at a rate represented by the associated failure rates. A state can be distinguished according to whether its failures were detected or not, and the corresponding transition rates are adjusted by the component coverage values. It is conservatively assumed that the *undetected failure* of a component *in use* results in a system failure.

At the first failure level all single failures causing system failure are aggregated to form the state SF-1. Analogously, all two-failure states which result in system loss are aggregated to create SF-2. The other two states in failure level two are P-mode and S-mode which are aggregates containing all two-failure combinations that result in these modes. For example, a combination of a detected failure of S1 on Channel A and a detected failure of the CPU on Channel B results in a transition to the S-mode since each channel's S1 sensor is no longer operationally accessible. A single trapping state is created at the three-failure level, state U-3. A conservative modeling assumption -- all third or greater failures are treated as a system failure -- is used.

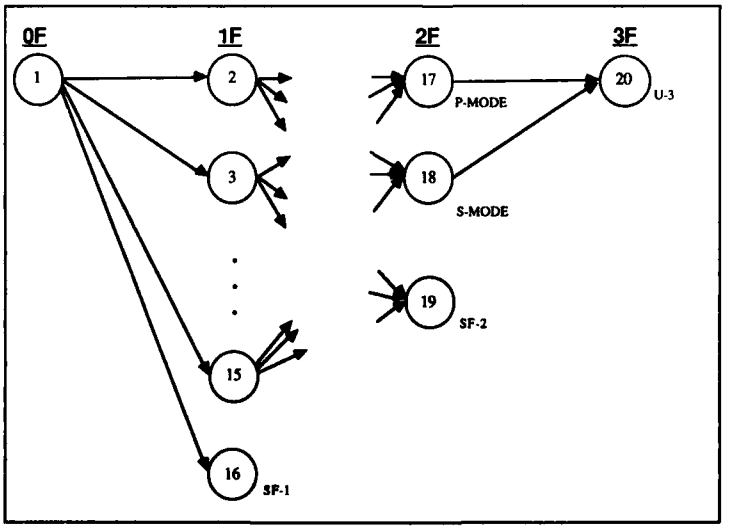



Figure 2. The Markov Model

Table 2. States in the Model †

| | | | |
|---|---|---|---|
| **FAILURE LEVEL 0** | | | |
| 1: | NO FAILURES | | |
| **FAILURE LEVEL 1** | | | |
| 2: | S1(A) – D | 10: | ACT(B) – U |
| 3: | S1(B) – D | 11: | CPU(A) – D |
| 4: | S1(B) – U | 12: | CPU(B) – D |
| 5: | S2(A) – D | 13: | CPU(B) – U |
| 6: | S2(B) – D | 14: | XLINK – D |
| 7: | S2(B) – U | 15: | XLINK – U |
| 8: | ACT(A) – D | 16: | SF–1 |
| 9: | ACT(B) – D | | |
| **FAILURE LEVEL 2** | | | |
| 17: | P–MODE | | |
| 18: | S–MODE | | |
| 19: | SF–2 | | |
| **FAILURE LEVEL 3** | | | |
| 20: | U–3 | | |

† D = detected failure, U = undetected failure

It should be emphasized that the model truncation at the third failure level introduces an approximation leading to upper and lower bounds on the probability of system failure. The lower bound is obtained by adding the probability of being in SF-1 and SF-2; the upper bound additionally includes U-3. Clearly the process of truncating the model reduces the amount of computational work required to construct and solve it, but it may also introduce an error that is unsatisfactory. This is determined by the discrepancy in the bounds. When the difference between the bounds is considered negligible, this Markov model is sufficient to predict the system's probability of system failure, thereby providing an analytical alternative to life-testing procedures.

Once the model states and transitions between them are established, the Markov model is solved as a first order differential system. The solution state vector gives the probability of being in each state of the model as a function of time.

3.1.2. Analyzing the Specific Configuration Objective.

From the system model of Figure 2, we proceed to analyze the specific configuration objective. Let the expression $Pr\{SF_i(\Delta T)\}$ denote the probability of system failure over a time period $\Delta T$ given that the system started with component i failed. *Notice that this is the probability of a conditional event in which the initial condition is the failure of component i.* Determining a formula for this probability is the key to satisfying the specific configuration objective, and a derivation of $Pr\{SF_i(\Delta T)\}$ is given in Appendix A.

The (average) rate to system failure for each component i, given that the aircraft is dispatched for a period $\Delta T$ with that component failed, is obtained from $Pr\{SF_i(\Delta T)\}$ by dividing by $\Delta T$. This rate is denoted by $SF_i(\Delta T)$. Plotting $SF_i$ as a function of the dispatch interval $\Delta T$ is illustrated in Figure 3. This figure represents the rate to system failure when operating with a failed CPU in Channel A. To understand why the initial position of the curve (at $\Delta T = 0$) is approximately $7.0 \times 10^{-5} hr^{-1}$, consider the fact that with the CPU(A) down, the subsequent failure, detected or undetected, on Channel B of the S2 sensor, actuator, or CPU brings the system down. In addition an undetected failure of the S1 sensor on Channel B contributes to system loss. Adding together these four failure rates from Table 1 gives $6.95 \times 10^{-5} hr^{-1}$.

When a constraint is enforced as illustrated by the horizontal line in Figure 4, one judges the dispatchability of a specific configuration by comparing its SF rate plot to the given target level. Clearly the time interval $\Delta T$ must be constrained so as to keep the SF rate below a designated level. In Figure 4 the upper bound on $\Delta T$ is 300 hours. If the target level is severe enough, a $\Delta T$ of zero may not be sufficient to comply. This means that the aircraft cannot be dispatched with that component failed, and the component is classified as non-dispatchable. From the SF rate plots of the system's components it is straightforward to compare and order the severity of component failures.[2] This ordering establishes in a systematic way the basis for an aircraft's minimum equipment list, and the basis for a policy of time-limited dispatch operation.

[2] An analogous evaluation is used to generate SF rate plots for multiple fault configurations. See [1].

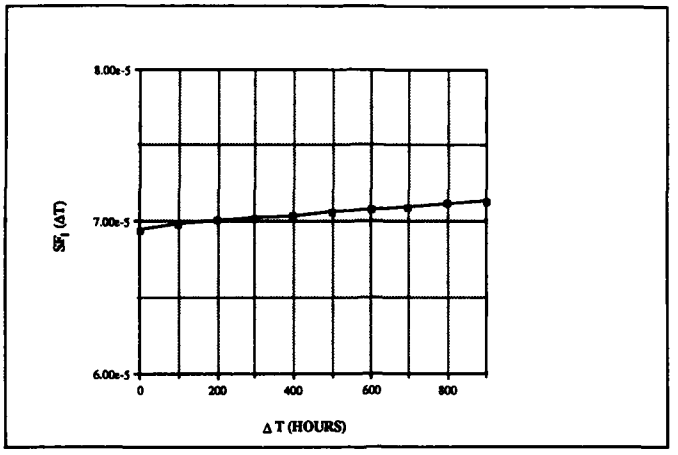



Figure 3. System Failure Rate when Operating with a Failed CPU

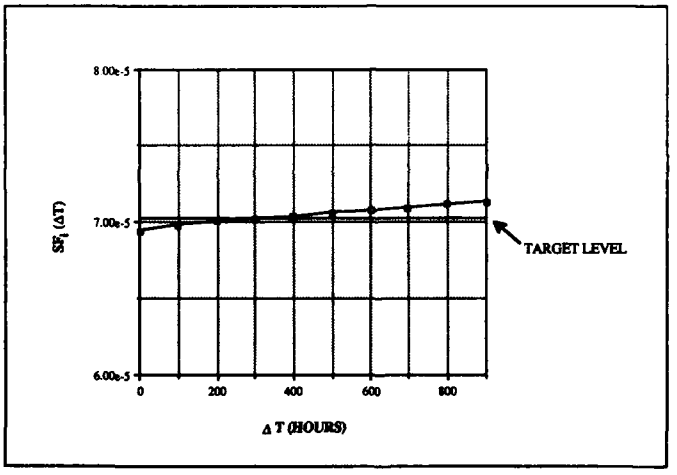



Figure 4. System Failure Rate when Operating with a Failed CPU.
Target Level Enforced

As an example, consider the ordering in Figure 5. Note that the components in SECTION 1 are significantly more vulnerable to SF than those in SECTION 2. In order of severity these components are:

| SECTION 1: | SECTION 2: |
|---|---|
| CPU(A) | S1(A) |
| CPU(B) | XLINK |
| S2(A) | S1(B) |
| S2(B) | |
| ACT(A) | |
| ACT(B) | |

The failure of a CPU generates the highest subsequent SF rate among the system's components; in other words, a CPU failure is a worse case single fault. Nevertheless, the aircraft can be dispatched for up to 300 hours following a single failure in the CPU as one determines from Figure 4. From this information, one can infer that from the occurrence of the first component failure, the system may be dispatched for up to 300 hours. That is, a 300 hour dispatch clock is set. Then, if a second failure occurs, but it is on the same channel as the first, the dispatch criterion remains unchanged; the 300 hour dispatch clock continues to run. This is because multiple faults on the same channel are no worse than a single failed CPU. If however, a second failure occurs on the alternate channel, the system dispatch clock may be reset to a much shorter time period, or the system may not be dispatchable at all. Finally, when a third failure occurs the system may not be dispatched until it is repaired to a full-up status. Thus, from the information provided by the component SF rate plots it is possible to devise a policy for time-limited dispatch operation.

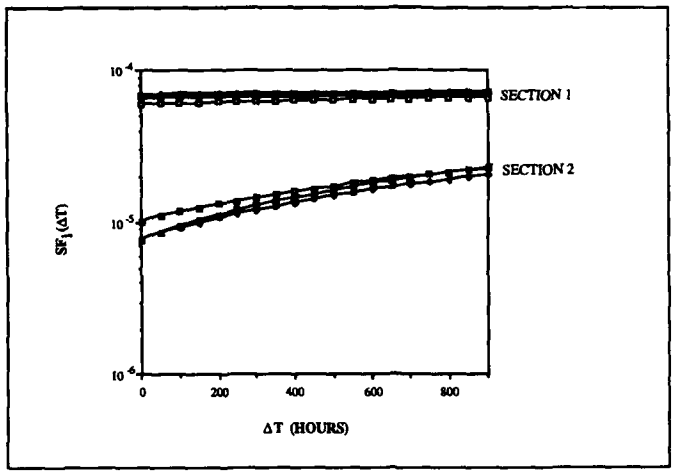



Figure 5. System Failure Rate Plots for all Components in the System

In conclusion, one observes that the time-limited dispatching operation is a function of two variables: failure configurations and dispatch time periods. As they vary, a family of dispatch criteria is generated. From these we must choose the best policy from the standpoint of control system reliability and economy. Moreover, these variables do not necessarily vary independently of each other due to prescribed target levels or other enforced constraints as illustrated above. Thus the task of formulating the system impact as a function of the dispatch policy variables and finding the optimal solution within the problem constraints is far from a trivial endeavor.

3.2. The System Impact Objective.

In this section we illustrate one way of analyzing the system impact objective in the presence of a single constraint. This constraint is placed upon the *fleet-average system failure (SF) rate*. The expression, fleet-average system failure rate, refers to the traditional method of assessing a system's failure rate whereby the number of system failures occurring throughout the fleet is accumulated for a designated time period and then averaged over the number of aircraft in the fleet. Traditionally, fleet-average statistics have been the source of data for assessing system reliability. Typically, the fleet is credited with a target SF rate, $\lambda_{TAR}$, for a given control system. The system's performance is considered acceptable if its fleet-average SF rate remains below $\lambda_{TAR}$.

While the fleet-averaging or life-testing method may be acceptable for gauging a system's reliability, it precludes a *predictive* assessment of how well the system will work under a particular mode of operation such as time-limited dispatch. As such, one must instead try to measure and predict the impact of time-limited dispatch operation within this context by analytically deriving an expression for the system's SF rate as a function of the dispatch time interval $\Delta T$. Using such an expression, the expected length of time $\Delta T$ that a control system with a known fault can be dispatched is determined by imposing the requirement that the time interval be small enough to keep the system SF rate less than or equal to the given target level, $\lambda_{TAR}$. In other words, the system impact objective is to determine whether time-limited dispatch operation is feasible within the constraint of a given target level.

To accomplish this objective, it is necessary to assess both the SF rate due to undetected component failures and the rate due to detected failures. This is because the control system is always vulnerable to undetected failures even though from the pilot's viewpoint the system appears to be operating in a full-up or no-fault state. The SF rate due to undetected faults is denoted by $\lambda_{UDF}$.

Because of the model truncation employed at the level of three failures, the true rate $\lambda_{UDF}$ must be approximated by lower and upper bounds for any given time period. For example, recall that the SF-1 state, State 16, at the first failure level is composed entirely of SF events due to undetected failures of components in use. Thus the contribution from State 16 is always included in evaluating $\lambda_{UDF}$. Similarly, a portion of the second failure level SF-2 state, State 19, is included to reflect pairs of undetected failures. For the upper bound, the three-failure level trapping state, U-3, or State 20, is included. Thus, for a given time period, T, the rate $\lambda_{UDF}(T)$ satisfies the inequality:

$$\frac{P_{16}(T)}{T} + \frac{P_{19}(T)}{T} < \lambda_{UDF}(T) < \frac{P_{16}(T)}{T} + \frac{P_{19}(T)}{T} + \frac{P_{20}(T)}{T}$$

where the left-hand sum is the lower bound value, and the right-hand sum is the upper bound value. Table 3 illustrates lower and upper bounds on $\lambda_{UDF}(T)$ for various time steps. The use of a time-varying rate is explored in full in [1]. For the purposes of this example, we approximate $\lambda_{UDF}(T)$ by using only the first order (linear) terms of the probability expressions, $P_{16}(T)$, $P_{19}(T)$, and $P_{20}(T)$. This is a reasonable approximation for short flight periods as evidenced in Table 3. Since $P_{16}(T)$ is the only expression with a linear term, it follows that this approximation to $\lambda_{UDF}(T)$ is the same constant value regardless of the value of T. That value is exactly the transition rate from State 1 to State 16 which is computed to be $7.64 \times 10^{-6} hr^{-1}$.

Table 3. Bounds on $\lambda_{UDF}(T)$

| T | Lower Bnd | Upper Bnd |
|---|---|---|
| 10.0 | 7.64e-6 | 7.64e-6 |
| 20.0 | 7.68e-6 | 7.68e-6 |
| 30.0 | 7.72e-6 | 7.72e-6 |
| 40.0 | 7.76e-6 | 7.76e-6 |
| 50.0 | 7.80e-6 | 7.80e-6 |
| 60.0 | 7.84e-6 | 7.84e-6 |
| 70.0 | 7.88e-6 | 7.88e-6 |
| 80.0 | 7.92e-6 | 7.93e-6 |
| 90.0 | 7.96e-6 | 7.97e-6 |
| 100.0 | 8.00e-6 | 8.01e-6 |
| 110.0 | 8.04e-6 | 8.05e-6 |
| 120.0 | 8.08e-6 | 8.09e-6 |
| 130.0 | 8.12e-6 | 8.13e-6 |
| 140.0 | 8.16e-6 | 8.17e-6 |
| 150.0 | 8.20e-6 | 8.21e-6 |
| 160.0 | 8.24e-6 | 8.25e-6 |
| 170.0 | 8.28e-6 | 8.29e-6 |
| 180.0 | 8.31e-6 | 8.33e-6 |
| 190.0 | 8.35e-6 | 8.37e-6 |
| 200.0 | 8.39e-6 | 8.41e-6 |

If the target level, $\lambda_{TAR}$, is greater than $\lambda_{UDF}$, then time-limited dispatch should be feasible. To understand why consider now the expected SF rate due to a detected component failure. Recall from the analysis of the specific configuration objective that for each component there is associated the probability of SF given that the system is dispatched with that component failed. This is computed from the expression $Pr\{SF_i(\Delta T)\}$ as a function of the dispatch time variable $\Delta T$. By multiplying each component's $Pr\{SF_i(\Delta T)\}$ by the frequency or rate with which that component fails (Table 1), one obtains the proportion of SF rate due to dispatching the aircraft in that specific failed-component configuration for a limited amount of time $\Delta T$. Then, the expected SF rate is the sum of the SF rate contributions from all the components and, in addition, the SF rate due to undetected failures, $\lambda_{UDF}$. It is this final sum which is compared to the target rate, $\lambda_{TAR}$.

To summarize this analysis in convenient notation, let $\lambda_i$ represent the frequency or rate with which component i fails. Then the product, $\lambda_i Pr\{SF_i(\Delta T)\}$, is the proportion of SF rate due to dispatching the system with component i failed, for a period of time $\Delta T$. Finally by summing over all components, $\Sigma\ \lambda_i Pr\{SF_i(\Delta T)\}$ and including $\lambda_{UDF}$, one can solve for the expected dispatch time $\Delta T$ that satisfies the equation:

$$\lambda_{TAR} = \lambda_{UDF} + \Sigma\ \lambda_i Pr\{SF_i(\Delta T) \qquad (3.2.1)$$

Solution values of $\Delta T$ for various $\lambda_{TAR}$ inputs are given in Table 4.

Table 4. Solution values of $\Delta T$ corresponding to $\lambda_{TAR}$ inputs

| $\lambda_{TAR}$ | $\Delta T$ |
|---|---|
| 7.64e-6 | 10.0 |
| 7.77e-6 | 50.0 |
| 7.91e-6 | 100.0 |
| 8.03e-6 | 150.0 |
| 8.16e-6 | 200.0 |
| 8.29e-6 | 250.0 |
| 8.41e-6 | 300.0 |
| 8.53e-6 | 350.0 |
| 8.65e-6 | 400.0 |
| 8.77e-6 | 450.0 |
| 8.88e-6 | 500.0 |
| 8.98e-6 | 550.0 |
| 9.10e-6 | 600.0 |
| 9.20e-6 | 650.0 |
| 9.31e-6 | 700.0 |
| 9.42e-6 | 750.0 |
| 9.51e-6 | 800.0 |
| 9.62e-6 | 850.0 |
| 9.71e-6 | 900.0 |
| 9.81e-6 | 950.0 |
| 9.90e-6 | 1000.0 |

Taking a different viewpoint, observe that the right-hand side of Eq.(3.2.1) is just a function of $\Delta T$ which outputs a SF rate for each input value of $\Delta T$. This relationship is plotted in Figure 6 and is referred to as a system impact plot since it gauges the impact of SF vulnerability as the dispatch time interval varies. As expected, the SF rate increases with increasing dispatch time. Notice, however, that when one applies the target level constraint of Figure 4, the system impact plot shows an expected dispatch time of at least 1000 hours. This time-limit interval is substantially higher than the one determined in the component SF rate plot of Figure 4. The reason is that in the system impact plot, each possible component failure has been averaged into the impact assessment according to its frequency, and the frequencies are very small numbers. On the other hand, the SF Rate Plot of CPU(A) is measuring only the SF rate vulnerability when dispatching with a failed CPU in Channel A. Thus, a policy for time-limited dispatch operation which is defined within the constraints of component SF rate plots is generally conservative with respect to the constraint placed on the fleet-average SF rate.

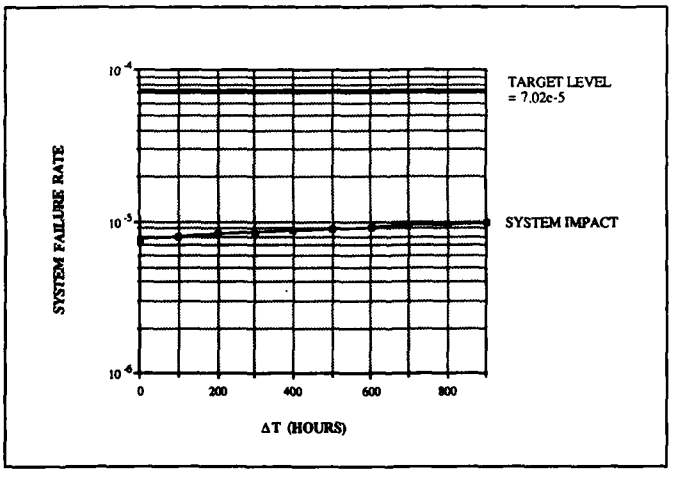



Figure 6. System Impact Plot

In concluding this section, it should be emphasized that the system impact objective can be interpreted in a variety of ways, one of which we have demonstrated here. Moreover, when the system is constrained by multiple operational goals, the task of determining system impact under a time-limited dispatch mode of operation can be a very challenging analytical problem [1]. Nevertheless, the pursuit of analysis and modeling methods to support this objective are absolutely essential to quantifying the effects of time-limited dispatch operation on the performance of a fault-tolerant flight control system.

Taken altogether the analytic tools and results developed in this paper provide the means for one to systematically and rigorously achieve the following goals:

1. Assess the feasibility of time-limited dispatch operation with respect to a given fleet-wide objective.
2. Determine the dispatch status of each system component or configuration.
3. Evaluate the effects of time-limited dispatch operation on the reliability of a flight critical control system.

## REFERENCES

1. Allinger, D.F., F.J. Leong, P.S. Babcock. *The Role of Markov Models for Analyzing the Time-Limited Dispatch Reliability of a Dual-Redundant Engine Control System.* C.S. Draper Laboratory, Inc., Cambridge, MA. December, 1987 (CSDL-R-2028).

## APPENDIX A. – A DERIVATION OF $Pr\{SF_i(\Delta T)\}$

To complete the illustration of how the specific configuration objective is analyzed we give a brief derivation of the expression $Pr\{SF_i(\Delta T)\}$ for a given failed component i. Because this is the probability of a conditional event, the relevant model configurations are submodels of the system model in Figure 2. These submodels are referred to as the time-limited dispatch reliability (TLDR) models and are illustrated in Figures 1A - 5A. For ease of discussion, suppose that the failed component under consideration is the S1 sensor on Channel A, S1(A). In Figure 1A this specific known failure and all the possible subsequent transitions that it induces are highlighted as a submodel of the system Markov model. Figure 2A illustrates this TLDR model, TLDR I, exclusively. Several observations are in order. First, the structure of TLDR I is the same for all components, only the transition rates $\lambda_{FAULT}$, $\lambda_{P\text{-}MODE}$, $\lambda_{S\text{-}MODE}$, and $\lambda_{SF}$ change according to the specific component fault. Secondly, the probability of a system failure while in this configuration is obtained by summing the probabilities of states SF-2 and U-3. As in the case of the system Markov model, this sum is actually an upper bound for the probability of system failure because U-3 includes all configurations of three or more failures, both failed and operational. The rate $\Sigma$ is chosen to be artificially large to insure a conservative analysis. Thus, all possible subsequent transitions stemming from a known failure have been accounted for in a conservative manner.

Since the TLDR I model extends to the second failure level, it induces a differential system that can be solved in closed form. In particular, the probabilities of SF-2 and U-3 as functions of $\Delta T$ are given by:

$$Pr\{SF\text{-}2(\Delta T)\} = \lambda_{SF}\frac{(1 - e^{-\Gamma\Delta T})}{\Gamma}$$

and:

$$Pr\{U\text{-}3(\Delta T)\} = (\lambda_{P\text{-}MODE} + \lambda_{S\text{-}MODE})\,\Sigma\,\frac{\left\{\frac{(1 - e^{-\Gamma\Delta T})}{\Gamma} - \frac{(1 - e^{-\Gamma\Delta T})}{\Sigma}\right\}}{\Sigma - \Gamma} \qquad (A.1)$$

where $\Gamma = \lambda_{P\text{-}MODE} + \lambda_{S\text{-}MODE} + \lambda_{SF}$. For short time periods, which are typically of several hundred hours, the expressions in Eq. (A.1) are well approximated by quadratics, namely:

$$Pr\{SF\text{-}2(\Delta T)\} = \lambda_{SF}\,\Delta T - \lambda_{SF}\,\Gamma\,\frac{(\Delta T)^2}{2}$$

and:

$$Pr\{U\text{-}3(\Delta T)\} = (\lambda_{P\text{-}MODE} + \lambda_{S\text{-}MODE})\,\Sigma\,\frac{(\Delta T)^2}{2} \qquad (A.2)$$

Thus, the probability of SF as a function of the dispatch time, $\Delta T$, in model TLDR I is approximately:

$$Pr_1\{SF(\Delta T)\} = \lambda_{SF}\,\Delta T + [(\lambda_{P\text{-}MODE} + \lambda_{S\text{-}MODE})\,\Sigma - \lambda_{SF}\,\Gamma]\,\frac{(\Delta T)^2}{2} \qquad (A.3)$$

There is one other configuration that must be accounted for in deriving the probability of SF given a failed component. This configuration is highlighted within the system Markov model as shown in Figure 3A. Even if the failure of the S1 sensor is the first detected failure, it may have occurred after the undetected failure of another component such as the S2 sensor on the alternative channel, Channel B. Figure 4A illustrates this TLDR model, TLDR II, exclusively. Both TLDR models, I and II, are shown in Figure 5A.

From TLDR II, the contribution to SF is conservatively estimated by the probability of U-3. Using a quadratic approximation, the probability of SF as a function of the dispatch time, $\Delta T$, in TLDR II is approximately:

$$Pr_2\{SF(\Delta T)\} = \Sigma\,\Delta T - \Sigma^2\,\frac{(\Delta T)^2}{2} \qquad (A.4)$$

The final step in this derivation is to assess the proportion of time that the dispatch configuration of TLDR I occurs versus the configuration of TLDR II. In other words, when a component such as the S1 sensor fails and is detected, is the true configuration given by TLDR I or TLDR II?

To determine these proportionality constants, one computes the probability of entering the FAULT states in both TLDR I and TLDR II. Let $IC_1$ denote the probability of the state, FAULT, in TLDR I:

$$IC_1 = Pr\{Fault\}_1$$

and analogously:

$$IC_2 = Pr\{Fault\}_2 \qquad (A.5)$$

Of course, $IC_1$ and $IC_2$ vary according to the specific component fault that is being evaluated, and $IC_1$ and $IC_2$ are really functions of time. But a constant value approximation to each of the ratios, $\frac{IC_1}{IC_1 + IC_2}$ and $\frac{IC_2}{IC_1 + IC_2}$ is obtained by using only the lower order terms of $IC_1$ and $IC_2$.

Now, the probability of SF expression for each TLDR submodel is weighted or multiplied by its proportionality factor and this completes the expression of the probability of SF given that a component has failed. In summary, let $\rho 1$ and $\rho 2$ represent the constant value proportionality factors, namely:

$$\rho 1 = \frac{IC_1}{IC_1 + IC_2}$$

and:

$$\rho 2 = \frac{IC_2}{IC_1 + IC_2} \tag{A.6}$$

For a given failed component i, the probability of SF over a time period $\Delta T$ is given by:

$$\Pr\{SF_i(\Delta T)\} = \rho 1 \left\{ \lambda_{SF} \Delta T + [(\lambda_{P\text{-}MODE} + \lambda_{S\text{-}MODE}) \Sigma - \lambda_{SF} \Gamma] \frac{(\Delta T)^2}{2} \right\} + \rho 2 \left\{ \Sigma \Delta T - \Sigma^2 \frac{(\Delta T)^2}{2} \right\} \tag{A.7}$$

This completes the derivation.

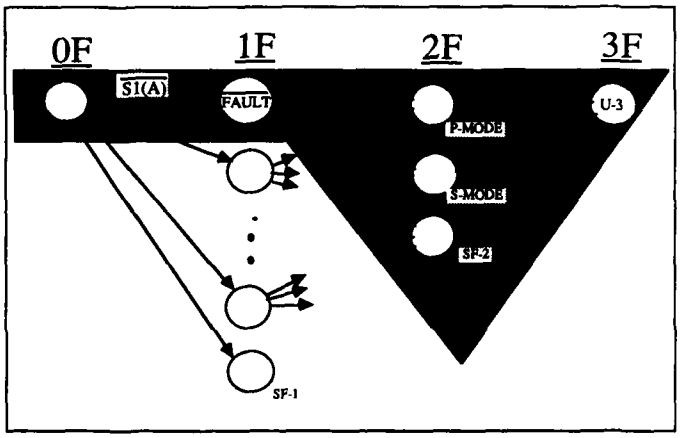



Figure 1A. Interesting Transitions in the Markov Model – 1

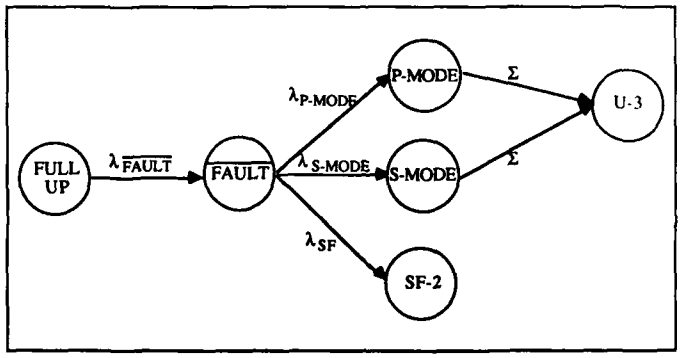



Figure 2A. Time-Limited Dispatch Reliability: Submodel I

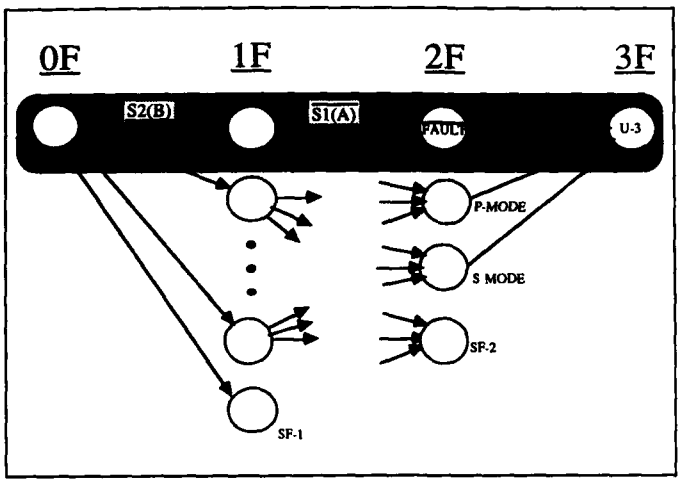



Figure 3A. Interesting Transitions in the Markov Model – 2

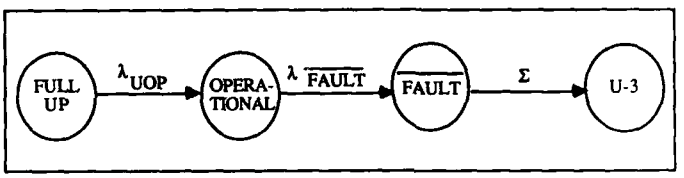



Figure 4A. Time-Limited Dispatch Reliability: Submodel II

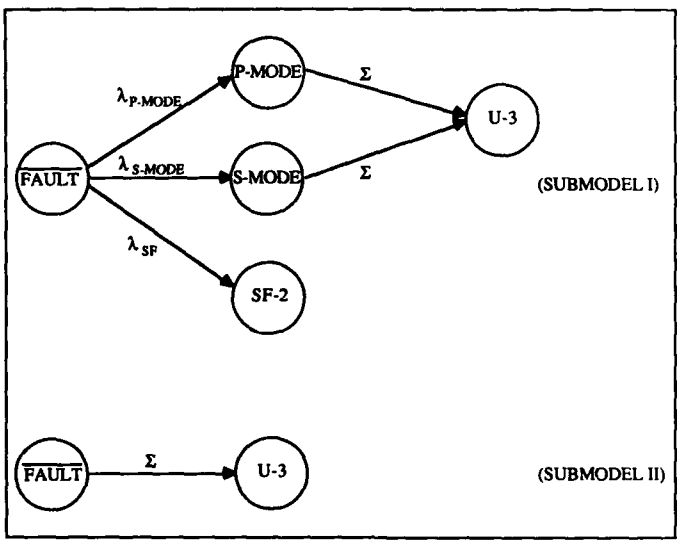



Figure 5A. Time-Limited Dispatch Reliability:
Submodel I and Submodel II for a Specific Component

# A FAULT TOLERANT FLY BY WIRE SYSTEM FOR MAINTENANCE FREE APPLICATIONS

R W DENNIS - Divisional Manager - Flight Controls Division
A D HILLS - Technical Manager - Flight Controls Division

GEC Avionics Ltd
Airport Works,
Rochester,
Kent, ME1 2XX
ENGLAND

**SUMMARY**

The paper describes a triplex primary flight computer system based on a reconfigurable architecture with extensive use of Application Specific Integrated Circuits (ASIC). The system is under development by GEC Avionics for Boeing Commercial Airplanes and comprises fault tolerant Fly-by-Wire (FBW) computers which are triplex dissimilar in both software and hardware. These command Actuator Control Electronics (ACE) units via DATAC (ARINC 629) data buses.

The Fly-by-Wire computers form the core of the full authority FBW system and perform all the computational commands for the pitch, roll and yaw surface actuation systems. The key requirements placed on the FBW computers are:-

* The probability of loss of the FBW function due to random failure in the FBW computer system shall be less than 1.0E-10.

* The FBW system shall be able to survive a generic failure which could arise from either hardware or software.

* The system reliability shall have a design aim of 0.95 dispatch probability after at least 30,000 operating hours.

The architectural design issues, in terms of integrity requirements and fault tolerance, are reviewed leading to a design which not only meets civil safety requirements but also has ultra highly reliability offering little or no maintenance action.

The FBW computer architecture is based on dividing the basic path into three sub-functional elements. Each of these elements is then replicated to provide fault tolerance. Communication between any one element and its adjacent elements is via point to point bidirectional serial data buses. For a FBW computer to be operable only one of each element type needs to be functional.

The internal element redundancy management function, performed both in hardware and software, is able to detect and isolate faulty elements and perform the necessary reconfiguration. Redundancy management is also addressed from a system viewpoint together with the implementation in terms of both hardware and software.

The development hardware produced is described, including the ASIC designs. The software structure and the use of dissimilarity is also addressed.

The Fly-by-Wire system is being evaluated by Boeing Commercial Airplanes using an iron bird rig in which FBW computers, DATAC buses, Actuator Control Electronics and actuators have been installed.

**INTRODUCTION**

In current commercial and military aircraft, avionics plays a key role in the utilisation of the airframe.

Mechanical assemblies such as the airframe and powerplant exhibit totally different availability characteristics from the avionics, the airframe having a failure rate, that to a reasonable approximation, increases with age i.e it wears out. Avionics, on the other hand, has a relatively constant failure rate typical of random component failure. Availability of the airframe and its mechanical assemblies can be improved by scheduled maintenance whereas avionics has required a completely different approach.

Availability is associated with life cycle costs, which in the commercial aviation environment emphasises the need for minimum maintenance down time and minimisation of costs associated with unscheduled departure delays due to equipment failure. In the military conflict environment it is more associated with the need for a sudden requirement and/or sustained use. Wartime and peacetime requirements are inherently different as during peacetime maintenance and preparation time are not as limited and flying hours are controlled to a lower level. The emphasis in peacetime is therefore on the ability to carry out a successful mission, or series of missions, at any time and in any place.

The availability of mechanical systems is increased through scheduled maintenance with the scheduled life being controlled to the extent, that with a high probability, it will continue to work satisfactorily for a known time. Availability of current avionics is based on the failure rate per hour of such equipment with the consequent fitting of the appropriate number of units to assure a single flight or mission success. This same failure rate is also used to calculate the number of spare units required to sustain a series of flight or missions over a given period of time.

The advent of fault tolerant avionics offers the potential to improve reliability and also integrity of key systems. In addition, availability can be greatly improved thereby reducing loss of revenue due to delays or AOG condition for the civil fleet and improve mission success rates for military aircraft. In addition fault tolerance enables dispatch following random failures provided a safe minimum level of assets are available enabling a shift from unscheduled to scheduled maintenance for avionics systems.

Military and commercial aircraft are becoming increasingly dependent on digital 'Fly-by-Wire' (FBW) technology where the pilot commands will be signalled electrically to the control surface actuation systems. This technology offers aircraft weight reductions, better fuel efficiency and has the potential for use of advanced control laws incorporating envelope protection features, performance and safety improvements.

This paper describes a Primary Flight Computer System (PFCS) for application to commercial aircraft although the fault tolerant architecture is also applicable to military systems. For the civil requirements the (PFCS) as the core of the FBW system must achieve a high level of reliability and integrity to meet the stringent safety requirements of the certification authorities. For example, the probability of loss of function must be less than 1.0E-10 per hour. This requirement for high integrity is met by the use of a fault tolerant architecture that is capable of surviving random hardware failures as well as generic hardware or software faults. In addition the system must provide a control path which endures beyond the minimum normal operating configuration. This "never give up" philosophy is important to ensure complete confidence for crews and passengers.

To achieve this integrity and reliability the techniques adopted are based on replication of the basic computing task to form redundant computing lanes. Inter lane redundancy management, based on output commands comparison, is then used to isolate the failed lane by a majority decision. Thus, in the general case, by adoption of this philosophy and if the system degrades gracefully, N-2 failures can be survived for an N lane system.

Recent research programmes have led to the design of fault tolerant systems based on distributed processing. Although this has enabled fault detection to be identified at sub-function level, the failure still invariably leads to a total shut down of a complete lane. An example of an advance distributed architecture is the MAFT (Multi Computer Architecture for Fault Tolerance) (1) which physically partitions the software tasks into "application" and "system overheads" processing.

The use of dissimilarity in hardware and software in redundant systems, has been previously successfully employed to circumvent generic failures. The benefits of this approach are based on the assumption that generic failures will occur at random and will be unrelated, thus the probability of two or more versions failing virtually simultaneously in a like manner will be extremely low. Examples of dissimilar hardware and software implementation are the A310, A320 secondary flight control systems (2 versions) (2, 3).

Reductions in component failure rates and new component packaging techniques such as Application Specific Integrated Circuits (ASIC), computerised thermal analysis techniques, developed to meet the changing installation environment, and advancements in screening techniques have further assisted the designer to make full use of the new technology in creating systems where reliability factors have begun to take on a totally new significance in system terms.

Furthermore, for aircraft emerging in the early 1990s, greater emphasis will be placed on operating costs, which must be made substantially lower than they are today. This means that in the area of avionics, unscheduled maintenance must be drastically reduced or totally eliminated. Thus, flight critical systems of the future must be designed to reduce life cycle costs and carry additional redundancy not just associated with safety aspects but to facilitate scheduled maintenance at maximum intervals. Although major advances in component technology have occurred in recent years, this improvement alone is considered insufficient to significantly enhance system reliability. If this increase in reliability is to be made, then the classical system architectures must be adapted to incorporate secondary redundancy, that is, each lane must be made fault tolerant to hardware failures.

In response to a Boeing Commercial Airplanes request, GEC Avionics embarked on a programme to develop the prototype FBW Primary Flight Computer System for future commercial aircraft. The fault tolerant architecture developed exhibits life cycle cost improvements over a system of conventional design and achieves high mean time between maintenance. The significant improvements in performance are made by use of a novel reconfigurable architecture for implementing the lane function together with intensive use of Application Specific Integrated Circuits (ASIC).

**PRIMARY FLIGHT COMPUTER SYSTEM DESCRIPTION**

The PFCS LRU configuration for evaluation on the 757 iron bird rig is shown in Figure 1 integrated into the FBW/L Electronic Flight Control System (EFCS).

There are three fault tolerant asynchronous Primary Flight Computers (PFC) which form the heart of the EFCS. Each PFC forms one independent digital computing lane of a triplex system architecture and is implemented with dissimilar hardware where necessary and programmed in dissimilar software. It receives data from the flight deck (control sticks, rudder pedals, trim switches) airdata, inertial reference systems, autopilot and other flight system sensors and computes commands for the pitch, roll and yaw surface actuation systems in order to provide the required flight control, stability augmentation and envelope protection functions.

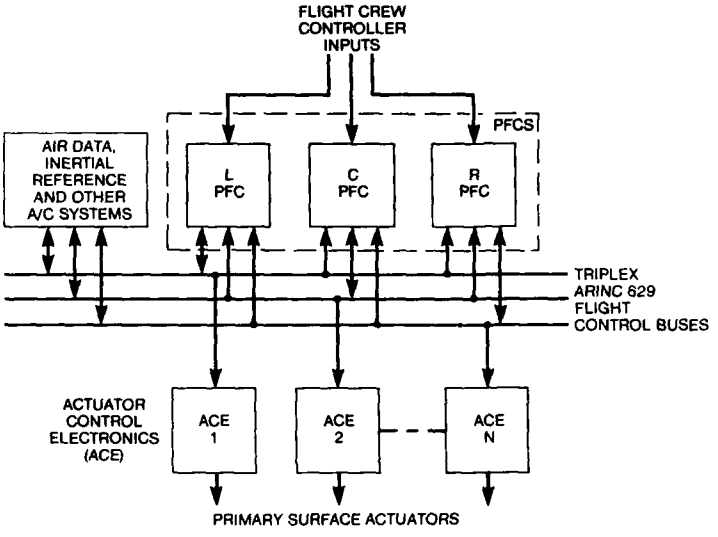



Figure 1 PFCS Configuration

The control stick and rudder pedal sensors are analogue and triplicated and are partitioned between the PFC to provide physical segregation and to reduce the overall hardware requirements per computer. Thus each PFC forms a command signal selection from a triplicated set, with one signal received directly and two received cross-lane. The command trim function is considered to have a lower availability requirement and consequently the associated control inputs are implemented as dual redundant discrete sensors partitioned between two PFC.

Except for the flight deck inputs and trim drive discretes, all other data exchanges between the PFCS and the other EFCS systems are accomplished via a triple dissimilar DATAC flight control bus. DATAC is a candidate for a new industry standard high speed digital serial bus (ARINC 629) and is based on carrier sense multiplex access protocol with collision avoidance. Currently the rig transmission standard is electrical using twisted pairs but optical bus structures are also under Boeing evaluation for future application.

Each PFC receives data from the three flight control buses but only transmits onto its associated DATAC bus, to protect against common mode transmission failures. The PFC are designated Left, Centre and Right to reflect the bus on which they transmit.

The Actuator Control Electronics (ACE) units provide the interfaces between the flight control buses and the aircraft surface actuation systems. Each ACE accepts control commands only from the PFC which transmits onto its associated bus and provides drive signals to a number of dedicated hydraulic actuators. This unit is assumed to be 'smart' and capable of monitoring its operation to the specified integrity level. Figure 2 shows a rack mounted ACE which is under evaluation on the 757 iron bird rig.

The PFCS forms the core of a full authority FBW system, the Electronic Flight Control System (EFCS), and performs all the computational commands for the pitch, roll and yaw surface actuation systems. The key requirements are:-

* the probability of loss of FBW/L capability, due to random failure in the PFCS, shall be less than 1.0 E-10.
* the system shall be capable of survival of a generic failure case which might be manifest either in the hardware or the software
* the system reliability shall have a design aim of 0.95 dispatch probability after 30,000 operating hours

Figure 2 Rack Mounted ACE

The safety and integrity requirements could be satisfied using a conventional design with two PFC each implemented with triplex dissimilar lanes. However, in order to achieve the desired reliability goal in a cost effective manner, significant secondary fault tolerant capability to random hardware failure was required.

The chosen reconfigurable architecture has the potential to achieve the very high reliability required when implemented using the recent advances in digital component technology. This approach has therefore, become the basis of the development programme undertaken by GEC Avionics.

**RECONFIGURABLE, REDUNDANT ARCHITECTURE CONCEPT**

Consider the reconfigurable redundant achitecture illustrated in Figure 3 where the basic lane function is partitioned into M elements each of which is replicated N times.

Assuming that:

- any element can transmit to and receive from the elements in the adjacent columns.
- the secondary or intra-lane redundancy management function facilitates lane operation down to one element of each column type.
- the failure rates ( $\lambda$ ) of the elements are identical.

then the probability (P) that the lane is functional after operating time (t) can be simplified to:-

$$P = (1- (1-e^{-\lambda t})^N)^M$$

Using this expression, the dispatch probability curves for two hardware configurations are plotted in Figure 4. Case A relates to a classical quadruplex redundant lane implementation (M=1, N=4) and for case B the lane is divided into 4 elements each replicated 4 times (M,N=4), the fault tolerant implementation.

The lane failure rate for curve A is assumed as 5.0 E-5 per hour, (this represents the failure rate of a typical FBW computer lane) and the failure rate for each element of the fault tolerant lane is taken as 1.5 E-5 per hour which includes 0.25 E-5 failure rate allowance for the inter element communication interfaces and any additional monitoring hardware necessary to support the redundancy management.

A system is considered to require no maintenance if P > 0.95, which for configurations A and B is reached after 12,800 and 27,300 hours respectively. The example reconfigurable architecture clearly demonstrates superior performance (over 100% improvement), at the expense of a small increase in hardware requirements. In order for this novel architecture to be realisable, a number of design issues must be considered.

Hardware Partitioning

Each element within the lane needs to be a stand alone, self contained functional block to simply the fault detection and isolation task. This clearly limits the number of element types which can be accommodated in the lane. Ideally the functional blocks should also operate autonomously and transfer data to the processing elements with minimum transport delay. Where data is output, by an element, to an external system, an independent and proveable mechanisation must be provided which is capable of selecting one of the element's multiple inputs as the source of output data.

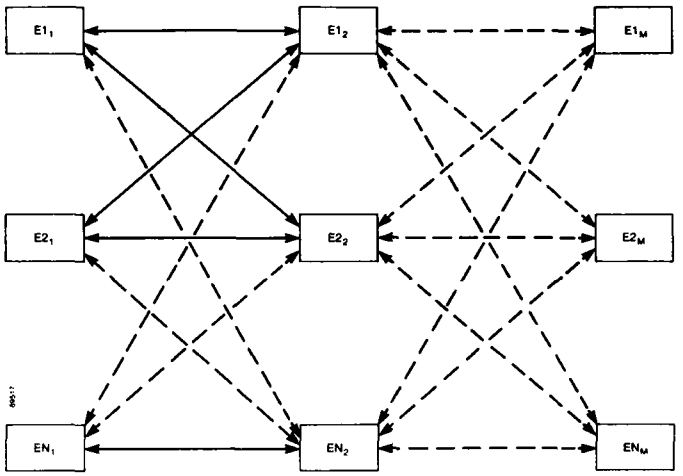



Figure 3 Fault Tolerant Architecture

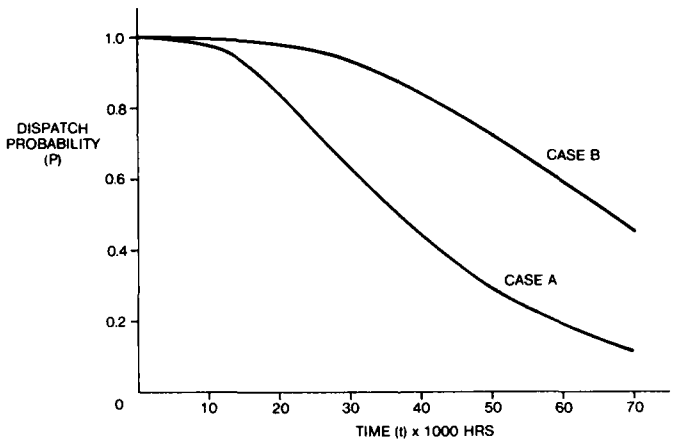



Figure 4 Dispatch Probability Curves

Data Transfer

The inter-element communication interfaces must provide adequate bandwidth capability and facilitate detection, isolation and containment of a communication path failure. To achieve data consistency and avoid "Byzantine Generals" problems (4), the transmission protocol must be "broadcast" and incorporate adequate error detection capability. The design should also lend itself to an ASIC implementation, to reduce parts count and therefore failure rate, and thereby maximise the performance of the reconfigurable architecture.

It is considered that these requirements are best served by a point to point high speed serial transmission system.

Interface Consideration

For elements which share common input/output devices, the design must preclude the propagation of faults between elements. Lightning and High Energy Radio Fields (HERF) induced transients present common mode hazards for signals which are shared by the replicated elements. Consequently, a robust approach must be adopted to protect against these effects.

Secondary Power Supply Architecture

With respect to power conditioning and distribution, two options may be considered, "consolidated" and "distributed". For the consolidated approach, the Power Supply Unit (PSU) forms an element which is replicated to provide failure survival capability. Each computing element is thus provided with multiple power inputs, one from each PSU. Methods for consolidating these inputs must be established and in the event of a PSU failure, the dynamic behaviour of the remaining working PSU must be considered.

The distributed approach entails fitting a dedicated power supply function on each redundant element. This approach appears to be simpler, but is only realisable if the failure rate can be kept low. Cost and reliability trade offs have been conducted for this option to establish the optimum approach for providing lightning and surge protection. A common protection network is preferred as against protection incorporated within each replicated supply element.

Redundancy Management

The internal redundancy management must detect and isolate a failure and perform the hardware reconfiguration. It therefore needs to be robust, analysable, proveable and able to differentiate between transient and hard faults. A total self monitored philosophy, applied at element level, leads to a simple redundancy management task and facilitates operation down to a single element. However, this implementation is considered technically high risk and requiring significant monitoring overheads with attendant cost penalty. A more flexible strategy based on a combination of monitoring techniques, ie, in-line, cross comparison and self monitoring leads to a more comprehensive fault coverage and at reduced risk and cost, noting that in a FBW application, redundant copies of input sensor data will be provided.

The redundancy management must also be able to resolve symmetrical failures, ie where, in a quadruplex architecture, two elements agree with each other but disagree with the other two. It must also address near coincident faults (5). The design must also reflect the "never give up" philosophy. For instance, if only two lanes of a triplex system remain but disagree, the system must continue to operate and make best use of available resources.

Since conventional test procedures cannot cover all aspects of the redundancy management design, new validation and verification procedures must be devised to facilitate design proving and hence certification. These are expected to encompass formal mathematical proof of the "core" redundancy management function including animation, simulation to provide rapid means of evaluating a large number of test cases, and "hands on" testing. In the latter case, response of the redundancy management to specific failure conditions can be investigated. This necessitates that a means must be provided which can independently inject faults into a previously good system.

**PFCS ARCHITECTURE**

The current rig standard internal PFC architecture is shown in Figure 5. Within each computer, the lane function is divided into three sub-functional elements or links, each of which is replicated 4 times. The links types are Peripheral, Processor and DATAC and each is contained on a printed circuit module. Communication between any one link and its adjacent links is via point to point bidirectional high speed serial digital paths. The resource requirements for the PFC to be operational is one healthy link of each type, thus the proposed architecture provides multiple survival capability of up to 3 failed links at each sub-function level.

The prototype PFC is powered by a single power supply module adapted in a manner to allow separate power control of each link and thereby model the "distributed" PSU configuration.

To complement the fault tolerant architecture, ASIC designs have been adopted to minimise the failure rate of each of the sub-functional elements. Further gains in reliability are made through the use of low power CMOS technology. The ASIC developed on this programme vary in complexity from 4000 to in excess of 14000 equivalent gates and utilise both gate array and standard cell CMOS technologies.

Each PFC is functionally identical but based on dissimilar microprocessors. In addition, it is intended that components which are not 100% testable or analysable be dissimilar in production applications. The need for dissimilarity at PFC level is driven by the requirement to survive a generic failure case such as a residual software error.

Each prototype PFC (shown in Figure 6) is currently contained in a 10 MCU ARINC 600 chassis and designed to be passively cooled while operating at a 65$^{o}$C ambient

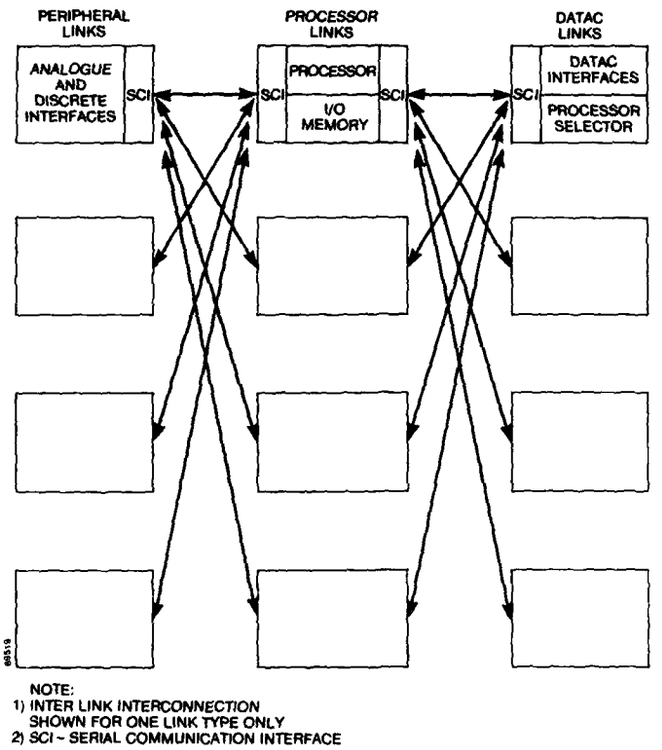



Figure 5 Prototype PFC Architecture

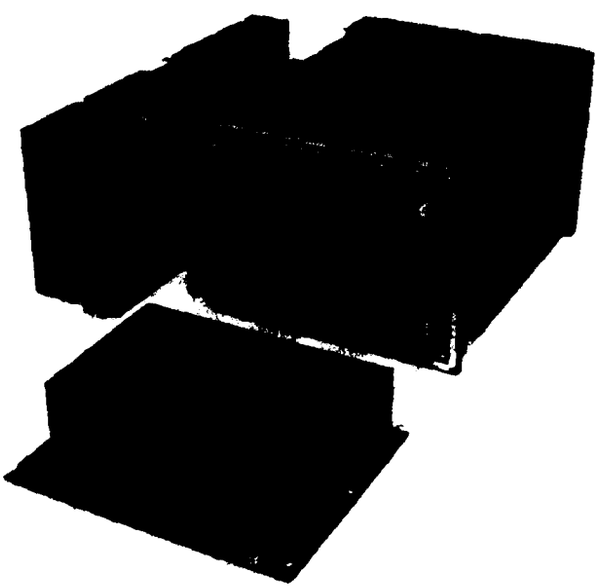


*Figure 6 The Primary Flight Computer*

temperature. The chassis was optimised to achieve good thermal performance, using computer modelling techniques and verified by tests conducted on a thermal mockup.

The Peripheral Link

This accommodates all the analogue and discrete interfaces for the PFC lane. The analogue to digital data acquisition system is autonomous and based on 12 bit conversion. Each digitised signal is simultaneously transferred to all the processor links via the Serial Communication Interface. Signals used in the forward computing path are serviced every 1ms to minimise transport delays, whilst the remaining signals are updated at lower rates. Extensive in-line monitoring and BIT circuitry is provided. This is used in the power-up sequence to augment the testing conducted by the PFC to establish the link operational status.

The Processor Links

These accommodate dual 32 bit microprocessor devices. Each link supports EEPROM based program store to provide in-situ reprogramming capability via the associated flight control DATAC bus, scratchpad memory, non-volatile fault store, watchdog monitor, CRC generator, memory decode and bus arbitration logic. The two quadruplex Serial Communication Interfaces are used for data exchanges with the peripheral and DATAC links. The data received via these interfaces is stored in a shared memory. The watchdog monitor requires a predefined sequence of checkwords to be written to it in a set period, otherwise all output transmissions from the Serial Communication Interfaces are inhibited. Discrete output signals from the link, which are independent of the Serial Communication Interfaces, provide DATAC and peripheral link shut down commands and processor link validity outputs. The three dissimilar processor elements with their associated languages are:-

a) Left PFC - Inmos Transputer T414/Occam
b) Centre PFC - Motorola 68020/Ada
c) Right PFC - Intel 80386/"C"

*The DATAC Link*

This provides the complete interface to the triplex flight control bus. Three DATAC terminals are contained on the module, two receive only and one receives and transmits, data for transmission being accessed from the memory resident in the Serial Communication Interface. The mechanisation used to select the processor as the source for control of the DATAC link and for writing data into this memory is incorporated in the Communication Interface. Data received by the terminals is simultaneously transferred to all processor links via the broadcast serial transmission paths. The received data is time stamped to facilitate testing for data refresh from the sourcing unit. The link also contains a cross lane inhibit function which shuts down DATAC transmissions if the link continues to transmit invalid command data after a predetermined period.

This function is intended as a final means of passivating a generic or software failure in the offending lane, wherein the local internal redundancy management may be inoperative or unable to detect and itself isolate the fault.

Communication Interface

The quadruplex Serial Communication Interface, Figure 7, is central to the PFC operation and handles and controls the data transfer between links. It comprises a single transmitter with 4 buffered output drivers and 4 independent receiver channels together with an interface to a parallel bus. It also contains memory for the temporary storage of data, plus control and monitoring functions for the internal data flow and external memory accessing.

One of the key functions executed by the Serial Communication Interface on the DATAC and peripheral links is selection of one of the receiver channels as the source for link control and PFC output data. The selection algorithm operates on status words provided by each processor link. This word contains a self opinion status bit based on the processor in-line monitoring plus inter-processor opinions derived from comparison of own and other processors output commands. The result of the selection process is echoed back to the processor links, which then compare the value against their own opinions. Thus an incorrect selection can be detected and the offending peripheral and DATAC link inhibited. All transmissions received from the processor links are checked for validity, using parity, word length, synchronisation period monitoring and frame time monitoring. If any monitor fails then the respective processor link is deemed to be faulty and not considered in the selection process.

**REDUNDANCY MANAGEMENT OVERVIEW**

An overview of the fault detection, isolation and reconfiguration strategy developed for the fault tolerant PFC follows.

All processor links within a PFC remain active. Each of these links contains an identical software suite and computes the total task, including control law processing, together with inter and intra lane redundancy management.

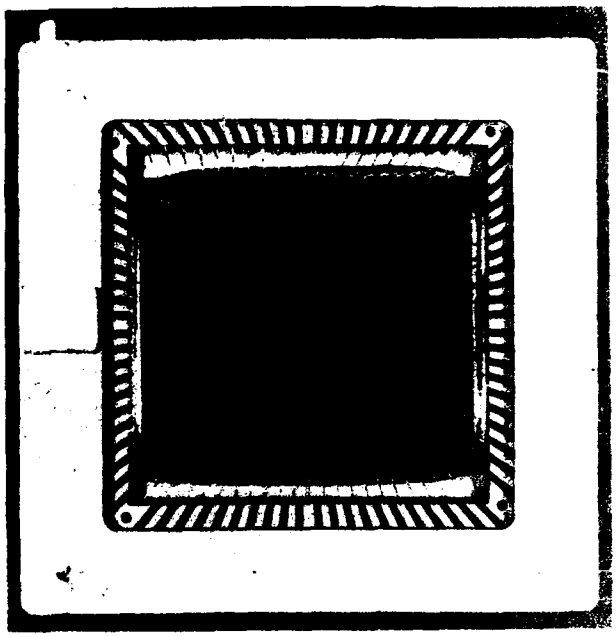


Figure 7 Serial Communication Interface (ASIC)

Inter PFC redundancy management is based on comparison monitoring of surface command outputs, the results of these comparisons being reflected in the status word transmitted as part of the message string on to the DATAC bus. This status word comprises, lanes own validity or "Available for Control" (AFC) flag, lanes opinions of the other two PFC validity and cross-lane threat and inhibit commands.

The validity (AFC) flag when asserted, indicates that this lane is available for actuator control. The state of this flag is derived from processor self-monitoring and comparison of own actuator command outputs with the outputs from the other two lanes. If the lane declares itself invalid, AFC will be cleared although command data will continue to be transmitted. If subsequently the lane tracks favourably for a period greater than twice the failure monitor delay, the AFC flag will be reasserted.

The lane opinion status bits are derived from the selected processor links opinion of the other lanes commands. This opinion is based on favourable and unfavourable comparison of cross-lane surface commands with respect to established thresholds and allowable delays.

The cross-lane threat and inhibit commands are used to inhibit DATAC transmission for an errant lane if it continues to output corrupt or erroneous data with its validity flag asserted.

The processor management is based on self assessment and cross processor monitoring. The self opinion is derived from, for example, power up BIT, failure history and in-line monitoring comprising frame overrun, software flow and watchdog monitors plus Serial Communication Interface wraparound and status checks. The cross processor opinions are derived from comparisons of actuator surface demands and active peripheral and DATAC link selections. Inter processor data transfers are achieved via the serial transmission nodes. The mechanisation used on the peripheral and DATAC links to select one of the four processors as the source of output data has been previously discussed. Symmetrical selection cases such as 2 against 2 processors are resolved in software, wherein the processors re-assess their self opinions by comparison of their surface commands with that of the other lanes.

The peripheral and DATAC links deemed to be healthy, can be in one of two states, "active" or "suspended". Only two links of each type will be active and thus monitored, one selected to form the computing path and the other placed in a standby mode. Suspended links are intentionally unmonitored to reduce the data handling and hence the computing necessary to perform the intra PFC redundancy management function. If in the event an active link fails so as to be considered condemned for the remainder of the flight, then one of the suspended links is re-instated to the standby mode to provide protection against a second failure. The task of allocating these states is performed on initial power up by consolidation of link availability opinions derived by each healthy processor based on completion of BIT and assessment of historical data on link 'health' held in non-volatile memory (NVM). The active links are cycled on each power up, thus ensuring

that each link is exercised at least every other flight to reduce time at risk for dormant failures.

The strategy for link condemnation and re-instatement is based on three classifications of failure.

- Currently condemned (CC) wherein the redundancy management has declared the link unavailable for use by the system but has not yet declared the link as condemned for this flight (CF).
- Condemned for this flight (CF) wherein power up BIT or redundancy management has declared the link as unavailable for duration of the current flight.
- Permanently condemned (PC) wherein the link is assumed to be failed permanently and can only be re-instated as a result of maintenance action.

The processor links operate in a frame synchronous manner from the same selected peripheral and DATAC links to minimise internal data skew.

The synchronisation establishment and maintenance algorithm is implemented in software and used to align the computing frames of the four processor links within a lane. Alignment of the computing frames is maintained to typically 50us. The algorithm is itself fault tolerant and can therefore cater for failed components of the system and is a derivative of that used successfully on the YC-14 and Jaguar FBW programmes.

**RELIABILITY**

The reliability of the PFCS has been predicted using analytical techniques. The reliability model comprises a 3 by 4 block matrix representing the failure rates of each link plus a single block which denotes all common mode failures. This fault affects all similar links. Software is assumed to be error free.

The system dispatch probability curve, shown in Figure 8, assumes a minimum dispatch condition (MDC) of at least two healthy links of each type in two PFC and one healthy link of each type in the remaining computer to satisfy the safety requirements. With the system at MDC and ignoring common mode effects, a minimum of three random failures must occur for the system to fall below the minimum operational configuration (two working PFC).

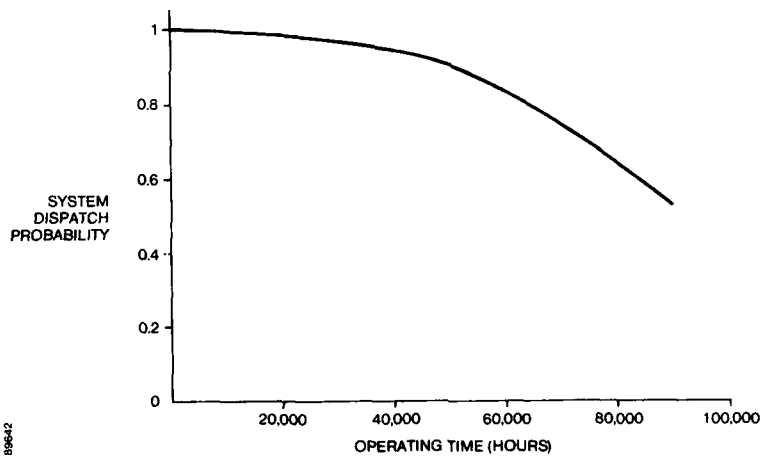



Figure 8 Dispatch Probability of the PFC

From the above curve, it can be seen that the 0.95 dispatch probability is reached after 38,100 operating hours. Thus the system is conservatively expected to be maintenance free for the first 7 years of in service life. (assuming the equipment operates 5,000 hours per year).

The mean time between mandatory maintenance action (MTBMMA) can be estimated by integrating the dispatch probability curve from 0 to ∞. However, as the aircraft life is expected to be in the region of 75,000 hours, a more realistic MTBMMA may be equated to the time to reach 0.5 dispatch probability. This is predicted as 92,700 hours per ship set.

To eliminate unscheduled maintenance, and any Aircraft On Ground situation, the PFCS can be programmed to provide a maintenance alert, for instance, when the system is one failure away from the MDC condition. This will give the airline operator ample time to plan the required maintenance action.

**SOFTWARE ARCHITECTURE**

The design methodology adopted, follows the classical route for development of dissimilar software suites for high integrity applications as defined in D0178A level 1. Each lane is developed by an independent team using dissimilar High Order Languages (HOL) to reduce common mode errors. From the requirements, each lane independently derives a lane-specific Software Requirements Document (SRD) and Software Structure Document (SSD), and follows the normal top down procedures to generate module design and code. Ada is used throughout as PDL and the top level design specified using a structured methodology. Testing is conducted at module, functional and lane level prior to the integration of the three suites at the system test stage.

The software is organised as 7 major functions illustrated in Figure 9. These are:

Executive
Input Signal Management (ISM)
Monitors and Test (MAT)
Fault Isolation and Configuration Management (FICM)
Control Law Processing
Output Signal Management (OSM)
Ground Maintenance BITE (GMB)

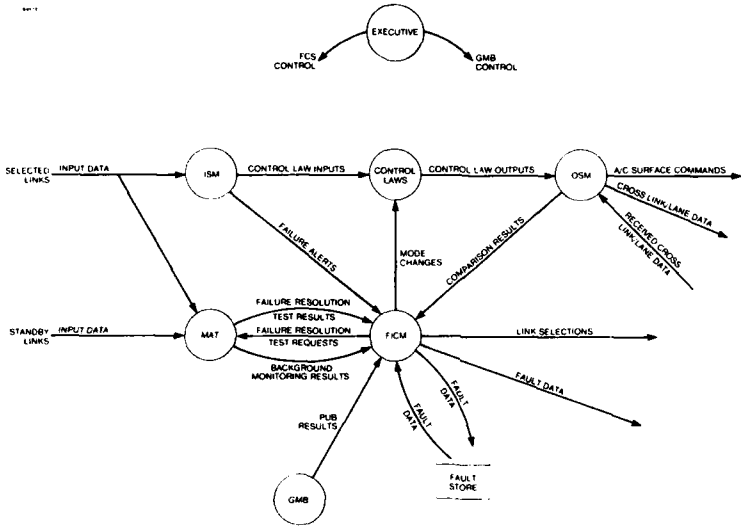



Figure 9 Software Overview

Software execution is based on a multi-iteration rate structure to optimise the conflicting requirements of control loop bandwidths, time delays and processor throughput loading. Forward path and inner loop computing is performed at 100Hz. The remaining processes are executed at lower iteration rates consistent with achieving the desired system performance.

The Executive contains functions to initialise, monitor and control the software and is responsible for such functions as updating the watchdog monitor, synchronisation and task scheduling.

On initial power up, the full complement of BIT tests are performed and coupled with assessment of the link failure history stored in non volatile memory, consolidated link availability status is established and the active link configuration defined for the flight. (At this stage, the dispatch assessment is undertaken across the complete system to ensure that the MDC is achieved) For an in air start up, the requirement is to bring the PFC on-line in minimum time. This is achieved by reconfiguring the redundant hardware to the same operational condition as existed prior to power down, and initialising integrators and filters to ensure rapid tracking.

ISM is responsible for conditioning and validation of raw input data and consolidating them with similar inputs from other lanes. It operates only on the selected peripheral and DATAC links and provides management for single, dual and triple, variable and discrete signals. The consolidation process is intended to isolate failed sensor signals from control law computing, minimising transients as necessary. Signals which fail the monitoring and comparisons are flagged to the FICM for fault identification and resolution. Included in ISM is a mechanism for integrator equalisation and mode consolidation. This is necessary at two levels, intra and inter PFC, to ensure that

individual processors within the PFC and other valid PFC, change mode simultaneously (on a majority vote) and achieve the desired tracking performance.

MAT consists of functions to monitor and test the local PFC and the external interfaced units. MAT monitors all active links "selected" and "standby" by judicious use of in-line, data comparison, data update, and wraparound check routines. MAT is not responsible for isolation of a failure, it is designed to conduct the tests and flag any failure to FICM for resolution.

FICM performs the main redundancy management function within the PFC. It acts upon fault data received from ISM, MAT and OSM and is responsible for failure isolation, internal reconfiguration and for invoking reversionary control law modes. It also consolidates status information for annunciation purposes and determines whether the PFCS satisfies the minimum dispatch criteria.

OSM provides three main functions. Cross-monitoring of processor output commands in order to generate opinions on the validity of each of the processor links; comparison of the own processor commands with those from other lanes in order to generate cross-lane opinions and validate the own lane's "AFC" status flag plus generation of output data to be used by other EFCS systems.

GMB provides PFCS BIT capability on power up, facilitates re-programming of the PFC code store via the flight control DATAC bus, and forms a maintenance aid to report on failures and enable selective PFC tests to be carried out by the maintenance crew to support LRU replacement verification checks. The ground maintenance function is only invoked if the safety locks which establish that the aircraft is "on ground" are asserted.

The proportion of the total software task, per software function, for one of the three dissimilar lanes is given below:-

| | |
|---|---|
| Executive | 3 % |
| Control Laws | 47 % |
| Inter Lane Redundancy Management | 12 % |
| Intra Lane Redundancy Management | 20 % |
| External PFC Redundancy Management | 18 % |

**757 IRON BIRD RIG EVALUATION**

Figure 10 shows the general layout of the 757 iron bird and Figure 11 shows the vertical fin and rudder arrangement in more detail. The 757 rig has been modified as follows:

a) The previous actuators have been replaced by modified units coupled to (ACE) units provided by three competing consortia, one of which is NWL/GEC Avionics

b) A triplex electrically signalled DATAC bus has been installed to enable PFCS, ACE and simulated Avionics communication.

c) The triplex PFCS have been installed

d) The rig enables evaluation of both dual and triple PFC/ACE/ actuator installations on various surfaces including force fight and the effects of the PFC asychronous operations.

To assist in this evaluation a sophisticated test set developed by GEC Avionics will enable the PFCS cluster to be exercised and failures to be induced. The test set also contains a simplified aircraft model to enable dynamic simulations and transients to be investigated

The rig is supported by a comprehensive monitoring test set up enabling actuator performance to be evaluated under all conditions from normal operation through transients to hard failures.

**RELEVANCE TO MILITARY SYSTEMS**

Fly by Wire technology has already found application in military aircraft, the United Kingdon Jaguar Fly by Wire demonstrator and EAP programmes having led to this technology being specified for future production programmes.

With respect to fault tolerance, this clearly has applications wider than flight controls. However, any system where high availability is a prime requirement should show clear benefits in terms of mission success rate.

The fault tolerant structure as described is flexible and has achieved a good balance between the use of fault tolerance and improvements to reliability of the building blocks through the use of LSI & ASIC technology. However, some aspects of the system design, such as the use of dissimilarity, while valid for commercial aircraft integrity requirements, may not be optimal for corresponding military applications where the integrity requirements are less stringent.

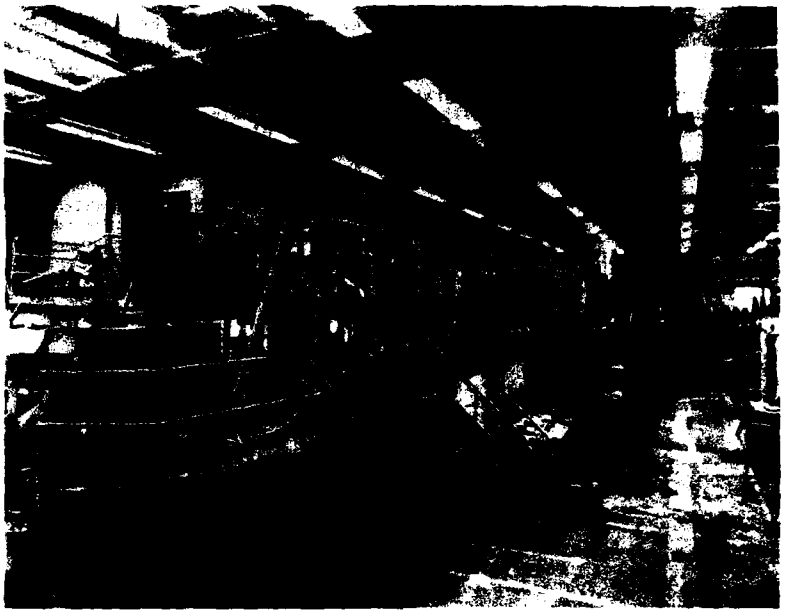


Figure 10 General Assembly of 757 Iron Bird

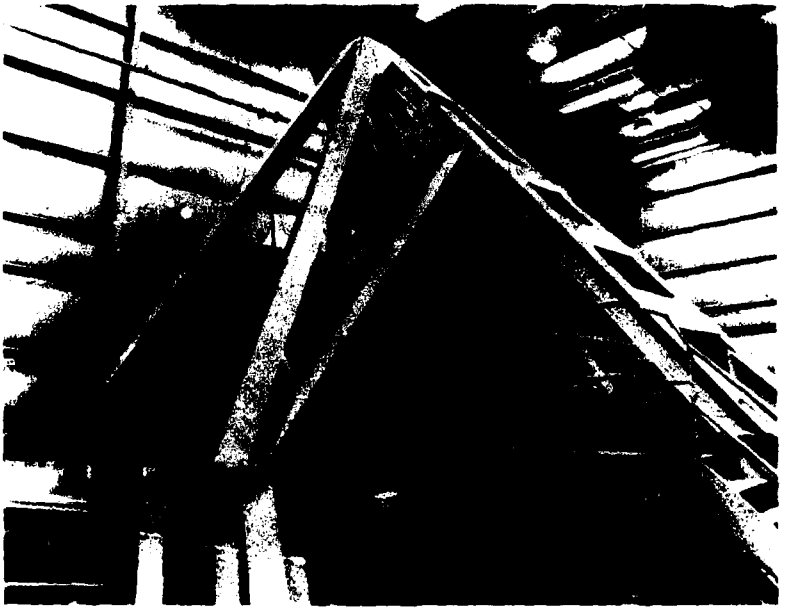


Figure 11 Rudder Arrangement of 757 Iron Bird

## CONCLUSIONS

The development programme undertaken by GEC Avionics and led by its Flight Controls Division has proven the viability of a fault tolerant primary flight control system based on secondary redundancy and extensive use of circuit integration.

The impact of Very Large Scale Integration (VLSI) upon fault tolerant architectures cannot be underestimated and the availability of powerful single chip microprocessors has revised the traditional design of avionic systems. Distributed processing can now be performed within an avionic function which has enabled the design of fault tolerant architectures such as the one developed by GEC Avionics.

The additional processing overhead for this advance is not inconsiderable but is within the capabilities of the current generation of 32 bit machines. In addition to the direct benefits gained from this programme a number of associated technologies have been matured. These include adaption of computer modelling techniques to enable trade studies of optimum levels of secondary redundancy to be conducted together with the generation of a model capable of simulating the asynchronous interaction of the PFCS and the associated DATAC buses for system tracking evaluation.

The necessity to verify our redundancy strategy has also required the design of a non-real time PFC computer model to permit the simulation of intra-PFC failure management techniques, the base strategy having already been proven on a transputer based demonstration unit.

The programme has shown that the goal of "fit and forget avionics" is now attainable, the extent of the reliability gain now and hence the reduced life cycle costs being balanced against the acquisition cost.

Acknowledgements

Acknowledgement is afforded to the associated GEC/Marconi Divisions, Combat Aircraft Controls Division, Power Conversion Systems Division, Flight Automation Research Laboratory and Software Systems Division without whose participation, this programme would not have been successful.

Thanks are also extended to Boeing Commercial Airplanes for their valuable support and contribution to this successful programme, including the use of the 757 rig photographs.

References

1. C.J. Walter, R.M. Kieckhafer and A.M. Finn, **"MAFT; A Multi Computer Architecture for Fault-Tolerance in Real-Time Control Systems"**. Proc. IEEE Real Time Systems Symposium, December 85

2. A.D. Hills, **"A310 Slat and Flap Control System Management and Experience"**. Proc. of Digital Avionics Conf. Seattle; Wash. Oct 31 - Nov 3. 1983

3. A.D.Hills, **"Digital Fly-by-wire Experience"** AGARD Lecture Series No. 143

4. L. Lamport, R. Shostak and M. Pease, **"The Byzantine Generals Problem"** ACM Trans. and Prog. Long. and Sys., Vol. 4 No. 3 pp 382-401, July 1982.

5. J. McGough, **"Effects of Near-coincident Faults in Multiprocessor Systems"**, Proc. of Digital Avionics Conf. Seattle; Wash., pp 16.6.1 - 16.6.7, October 31 - November 3 1983.

# The Integrated Airframe/Propulsion Control System Architecture Program (IAPSA)

Daniel L. Palumbo; NASA Langley Research Center; Hampton, VA 23665-5225
Gerald C. Cohen; Boeing Advanced Systems; Seattle, WA 98188
Charles W. Meissner; NASA Langley Research Center; Hampton, VA 23665-5225

## Summary

The Integrated Airframe/Propulsion Control System Architecture program (IAPSA) is a two-phase program which was initiated by NASA in the early 80's. The first phase, IAPSA I, studied different architectural approaches to the problem of integrating engine control systems with airframe control systems in an advanced tactical fighter. One of the conclusions of IAPSA I was that we had the technology to construct a suitable system, yet our ability to create these complex computer architectures has outpaced our ability to analyze the resulting system's performance. With this in mind, the second phase of IAPSA approached the same problem with the added constraint that the system be "Designed for Validation". The intent of the design for validation requirement is that validation requirements should be shown to be achievable early in the design process. IAPSA II has demonstrated that despite diligent efforts, integrated systems can retain characteristics which are difficult to model and, therefore, difficult to validate.

## Introduction

The Integrated Airframe/Propulsion Control System Architecture program (IAPSA) is a two-phase program which was initiated by NASA in the early 80's. The first phase, IAPSA I, studied different architectural approaches to the problem of integrating engine control with airframe control in an advanced tactical fighter [1] [2]. This effort was led by two prime contractors, Boeing Military Airplane Company and Lockheed-California Company, and was completed in 1983. One of the conclusions of IAPSA I was that we had the technology to construct a suitable system, yet our ability to create these complex computer architectures out paced our ability to analyze the resulting system's performance. This outcome came as no surprise to those who have been concerned with validating flight critical computer systems.

With this in mind, the second phase of IAPSA approached the same problem with the added constraint that the system be "Designed for Validation." The intent of the design for validation requirement is that validation requirements should be shown to be achievable early in the design process. By doing this, costly, and sometimes irrevocable, design decisions are avoided. The highest level requirements called for a safety requirement of $10^{-7}$ failures/hour, a mission requirement of $10^{-4}$ failures/hour and 100 percent system performance growth margin. The prime contractor for IAPSA II, Boeing Advance Systems, responded with what they termed a "Pre-Validation Methodology" [3]. In the Pre-Validation Methodology, thorough analyses of system reliability and performance is placed between system conception and final design, figure 1.

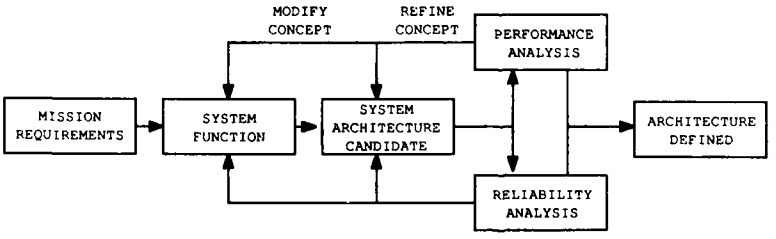



**FIGURE 1. PREVALIDATION METHODOLOGY**

### *Requirements*

The requirements were derived from an advanced, twin-engine, high-performance aircraft. The effort began with a complete requirements analysis of the proposed aircraft and expected mission scenarios. The performance requirements analysis resulted in a set of tasks and their associated processor and I/O demands (such as throughput, memory requirements and I/O bandwidth). Complete tables of this information were constructed. When totaled, the complete integrated system was projected to require from 0.5 megabytes (Mb) to 2.0 Mb of memory and from 0.5 million instructions per second (MIPS) to 4.0 MIPS of processing power. To complete the reliability analysis, each task was assigned to one or both of the reliability categories, mission success and flight safety, depending on their criticality (see table 1).

## Baseline Architecture

The Advanced Information Processing System (AIPS) was chosen as the component base from which the system was designed. AIPS is a NASA program which is producing a suite

of building blocks for constructing distributed fault-tolerant computer systems. Primary building blocks are Fault-Tolerant Processors (FTP's), network nodes and links, network management software, and device interface units (see figure 2).

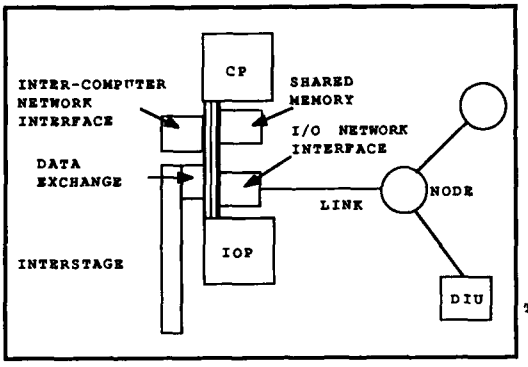



FIGURE 2. AIPS BUILDING BLOCKS

| SYSTEM FUNCTION | FAILURE EFFECT |
|---|---|
| MANUAL CONTROL | FLIGHT SAFETY |
| FLUTTER CONTROL | MISSION SUCCESS |
| TRAJECTORY FOLLO. | MISSION SUCCESS |
| WING CAMBER | MISSION SUCCESS |
| TRIM CONTROL | MISSION SUCCESS |
| INLET CONTROL | MISSION SUCCESS |
| ENGINE CONTROL | FLIGHT SAFETY |
| NOZZLE CONTROL | MISSION SUCCESS |

TABLE 1. FUNCTION RELIABILITY CLASSIFICATION

The FTP can be configured as a quad, triplex, dual or simplex. As shown in figure 2, the redundant channels of the FTP are connected through a data exchange device. The data exchange has multiple, cross-strapped channels which are designed so that they are isolated from each other and the FTP. The intent of this design is to provide protection against byzantine failures during exchange of single source data [4]. The data exchange is accessed through a controller which, among other things, provides a majority vote function. A channel of the AIPS FTP employs 2 processors, one computation processor (CP) and one I/O processor (IOP). The two processors are connected to a shared bus. Also on the shared bus is the data exchange device, scratchpad memory and I/O network interfaces.

The I/O network is composed of links, nodes and device interface units (DIU). Current AIPS technology uses twisted pair links which are operated at 2Mhz. A node is a circuit switch device and provides 5 full duplex ports. The DIU connects an I/O device to a link. During fault-free operation, the network is configured as a time multiplexed bus. The switching state of each node remains constant. A start-up algorithm ensures that all DIUs are accessible. When a network failure occurs and errors are detected, the defective component (a node or link) is located and isolated from the remaining network. New links are then enabled to restore lost capability.

It was originally expected that a network repair would be fast enough so that a single network could service the system. However, when it came time to layout the baseline architecture, the estimates of network repair time were so high that it was decided to provide 2 networks for each FTP. The baseline system, as depicted in figure 3, consisted of 3 FTPs. A single quad FTP computed the flight control laws and managed the two flight control networks. Two triplex FTPs were allocated for engine controllers (one to each engine). Each engine control FTP managed 2 engine networks. The 3 FTPs were connected by a triply redundant Inter-Computer network. The redundancy on this network is needed to maintain the reliability of data produced by the triplex and quad FTPs.

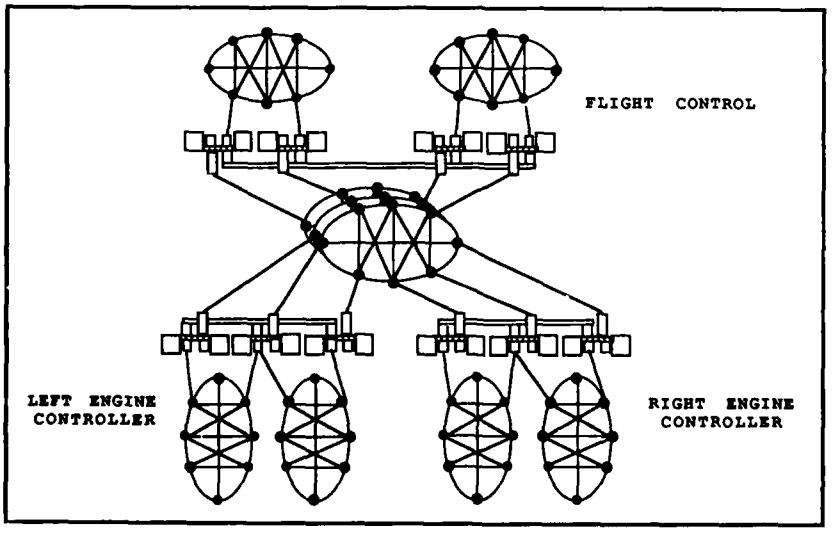



FIGURE 3. BASELINE CONFIGURATION

A flight control network contained 18 nodes, 46 links and 16 DIUs. Each network was connected to 3 FTP channels. One flight control network was linked to FTP channels 1, 2, and 3 the other network was linked to channels 2, 3 and 4. Only one channel controls the network at any time. Upon startup, channel 1 would control network 1 and channel 4 would control network 2. A single engine network had 5 nodes, 11 links and 3 DIUs. Each network was connected to only two channels of the triplex FTP engine controller.

## Reliability Analysis

The ASSIST/SURE [5] [6] reliability tools were used to do the reliability analysis. These tools were chosen because they are general and concise. SURE (Semi-Markov Unreliability Range Evaluator) uses an algebraic method to compute upper and lower bounds for a Semi-Markov model. The models are Semi-Markov because they allow the user

| SYSTEM PARTITION | PROBABILITY $\times 10^{-7}$ |
|---|---|
| FLIGHT CONT. SENSING | |
| PILOT | 0.0023 |
| BODY MOTION | 5.08 |
| AIRFLOW | 0.0078 |
| FLIGHT CONT. COMPUTER | 0.012 |
| FLIGHT CONT. SURFACES | |
| PITCH | 0.19 |
| ROLL | 0.38 |
| YAW | 0.19 |
| HYDRAULIC POWER | 0.036 |
| DUAL PROPULSION CONT. | 0.0076 |
| TOTAL | 5.9 |

TABLE 2. FLIGHT SYSTEM RELIABILITIES

| SYSTEM PARTITION | PROBABILITY $\times 10^{-4}$ |
|---|---|
| FLIGHT CONT. SURFACES | |
| PITCH | 0.18 |
| ROLL | 0.36 |
| YAW | 0.18 |
| PROPULSION (per engine) | |
| FIXED INLET | 0.046 |
| FIXED GUIDE VANES | 0.031 |
| ENGINE CORE SENSE | 0.00066 |
| CORE FUEL METER | 0.016 |
| AFTERBURNER METER | 0.076 |
| FIXED NOZZLE | 0.045 |
| ENGINE COMPUTATION | 0.0015 |
| PROPULSION DIU FAULT | 0.11 |
| AIRCRAFT TOTAL | 1.4 |

TABLE 3. PROPULSION SYSTEM RELIABILITIES

to express non-exponential recovery transitions in terms of the mean and variance of the recovery transition's distribution. The tool is fast and accurate. The ASSIST (Abstract Semi-Markov Specification Interface to the SURE Tool) tool provides a high level pascal-like language for specifying the models for SURE.

As was anticipated, it was found that the IAPSA system generated models which were too large to solve (due to state space explosion). The system was then partitioned. The contribution to failure for each section was computed using ASSIST/SURE and the results totaled, tables 2 and 3. As can be seen in table 2, the flight control system did not reach its reliability goal of $10^{-7}$ due mainly to the body motion sensors. This failure mode has been termed temporary exhaustion. The temporary exhaustion failure mode is a two step process. First, one of the four sensors or the DIU or link attached to that sensor, fails. The remaining sensors now function as a triplex. Then, if a failure occurs in the second network, two channels of the triplex set of sensors are temporarily "failed" while the network is repairing. This has been defined as system failure.

A second reliability analysis was done to compare the effectiveness of the mesh network concept versus a quad bus. The mesh network concept has been controversial in that it had never been established that the mesh network configuration produced any real gains in reliability over the quad bus. The analysis showed that, in fact, the mesh network produced practically the same reliability as the quad bus (The mesh might produce better availability figures, but this analysis has not been done). With the lack of demonstrable gain, it becomes difficult to justify the cost of validating the extremely complex network management software.

## Performance Analysis

The Discrete Event Network (DENET) simulation language was used to do performance analysis of the baseline IAPSA configuration. Timing and logic models of the AIPS building block hardware and software were constructed and used to extract data relating to scheduling behavior, network repair time and resource utilization. Figure 4 is an example of the type of data that can be extracted with a performance tool. The first two time lines represent utilization of the computation processor (top line, CP) and the I/O processor (second line, IOP). The last line displays utilization of the two networks (the networks have identical utilizations when fault free). The processing sequence begins with the IOP initiating the I/O for the 100Hz rate group. While the IOP waits for I/O completion, the FDIR task fires (see below for discussion of the FDIR task). Upon completion of FDIR, the IOP can complete processing of the I/O transaction.

When I/O processing for the 100Hz data is complete, the CP begins processing the 100Hz rate group while the IOP simultaneously begins I/O for the 50Hz rate group. The first experiment, which related to task scheduling phasing, resulted in unanticipated results and provides good insight into the challenges of validating advanced integrated systems.

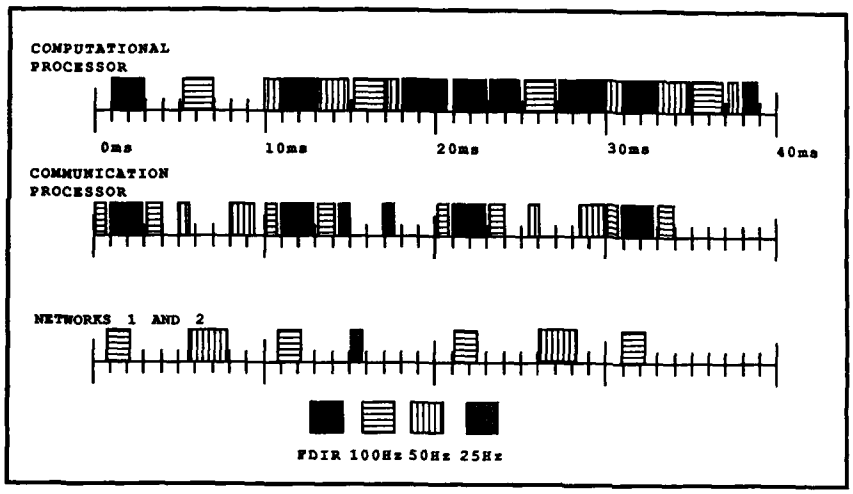



**FIGURE 4. PERFORMANCE DATA**

The FTP task scheduler is a priority based preemptive scheduler with three rate groups and a background slot. The three rate groups selected for the IAPSA study were 100Hz, 50Hz and 25Hz. Included in the 100Hz rate group is the fault detection, isolation and reconfiguration (FDIR) task. The FDIR task is part of the FTP's fault recovery process. The FDIR task has the highest priority and must run at the fastest rate to ensure high reliability.

| PHASE | 100 Hz | 50 Hz | 25 HZ |
|---|---|---|---|
| 0 | 2.93 | 7.01 | 15.54 |
| 1 | 2.7 | miss | 10.19 |
| 2 | 2.7 | miss | 10.19 |
| 3 | 2.7 | miss | 10.19 |
| 4 | 3.37 | 0.32 | 10.32 |
| 5 | 0.51 | 1.29 | 10.27 |
| 6 | 0.65 | 7.85 | 9.90 |
| 7 | 0.55 | miss | 10.19 |
| 8 | 0.85 | miss | 10.19 |
| 9 | 1.91 | 0.63 | 11.53 |

**TABLE 4. RATE GROUP DEADLINE MARGINS**

In what appears to be a rather mundane decision, The system designer must assign a start time within the 10ms time frame for the FDIR task. To provide some rationale for this decision it was decided to perform 10 runs of the simulation with the FDIR task scheduled to start at 10 different times within the 10ms time frame (0,1,2...9ms). It was decided to use the schedule which produced the minimum deadline margins. The results are shown in table 4. As can be seen, the 50Hz rate group missed deadlines when the FDIR task was scheduled at the 1,2,3,7 and 8ms slots. What is more indicative of the problem is that for start times of 0 and 6 the 50Hz rate group had a quite comfortable 7ms margin.

This wide of a range of values was not expected. When the analyst attempted to devise a statistically significant test that would validate that the deadline margins were maintained, he learned of the difficulty of predicting deadline margins in systems which employ priority based preemptive schedulers. Although this limitation is well known to some of those who study scheduling processes, evidently many system designers are capable of implementing this technology without a complete understanding of its implications. This is precisely the kind of problem which the design for validation philosophy was meant to identify.

## An Observation

It appears that a crossroads has been reached in aircraft control systems design methods. Historically, control law designers have placed a requirement on the lower

system level design that any fault tolerance be transparent to the control law application. In an integrated application such as IAPSA, this is clearly not possible. Conflict arises between the application's sensor redundancy management routines and the underlying system's redundancy management. Either the responsibility for managing I/O redundancy (including communication paths) must be completely delegated to the applications or it must be acknowledged that the system level fault tolerant functions cannot be effectively hidden from the applications.

Take, for example, these three cases relating to the handling of a quad redundant set of static pressure sensors: a failure free case, a case in which the sensor itself has failed and a case where some part of the redundant communications subsystem linking the sensor to the processor has failed. In the failure free case, the system delivers the four values intact to the application to perform its redundancy management. In the second case, when one of the sensors has failed, the fault tolerant system, being unable to detect an erroneous sensor value, delivers the one failed and three good values to the application. The application then takes appropriate action, first masking and then reconfiguring its known good sensor set. In the third case, in which a failure in communications occurs which causes a loss of the same sensor's value, the application, knowing nothing more, has to assume a failed sensor and reacts as in case 2. However, the fault tolerant operating system will also detect the failure and begin to take action to restore the communications path with the likely result that at some later time good values of the sensor will once again be available. Will the application then re-admit this sensor? If it doesn't, it will prematurely deplete the sensor set. If it does, it will need a good working knowledge of the underlying fault tolerant system's architecture and redundancy management functions.

## Conclusion

Two important lessons have emerged from the IAPSA II program. The first is that by adopting a design for validation strategy, costly design errors can be identified early in the design process. The pre-validation exercise also forces the designer to develop a better understanding of the limitations of the analytical techniques that must be employed to validate the system.

The second lesson is that the limitations of the analytical techniques can be too restraining. In the IAPSA II program, the computer based tools that were used had to be coerced into providing solutions. The user interfaces are primitive, data extraction is tedious and model correctness is always suspect. The IAPSA II problem, although not including any cockpit or armament subsystems, is an enormous problem. The interdependencies that are created between subfunctions when the subfunctions are integrated make it difficult to partition the problem. In order to partition the system, the analyst must make some assumptions about subsystem independence. The effect of the partitioning can be subtle and often introduces an unknown amount of error into the analysis.

## References

[1] Stern, A.D.,et al., National Aeornautics and Space Administration, "Study of Integrated Airframe/Propulsion Control System Architectures, (IAPSA) Volume II - System Requirements and Development", NASA CR 172174, October 1983.

[2] Bangert, L.H.,et al., National Aeornautics and Space Administration, "Study of Integrated Airframe/Propulsion Control System Architectures", NASA CR 172167, Nov. 1983.

[3] Cohen, G.C.,et al., National Aeornautics and Space Administration, "Design/Validation Concept for an Integrated Airframe/Propulsion Control System Architecture (IAPSA II)", NASA CR 178084, June 1986.

[4] Pease, M.,et al., "Reaching Agreement in the Presence of Faults", Journal of the ACM, Vol. 27, No. 2, Apr. 1980.

[5] Johnson, S.C., National Aeornautics and Space Administration, "ASSIST User's Manual", NASA TM 87735, Aug. 1986.

[6]*Butler, R.W. and White, A.L., National Aeornautics and Space Administration, "SURE Reliability Analysis", NASA TP 2764, March 1988.

# DEPENDABLE SYSTEMS USING 'VIPER'

by

**J.Kershaw**
Royal Signals and Radar Establishment
St. Andrews Road
Malvern, Worcs WR14 3PS
United Kingdom

## INTRODUCTION

Computer systems are being used increasingly in applications where a malfunction could cause loss of life or massive environmental damage. Redundancy (with 2, 3, or even 4 channels) is used to guard against random hardware failure in such systems, but redundancy alone does not protect against design faults which might affect every channel at the same time: the classic "Common-Cause Failure" (CCF). The risk of CCF is all-pervasive: from routine power failures, through physical mishaps such as dropped fire extinguishers to subtle effects arising from common maintenance procedures. Latent faults can be inserted, over a period, into all the channels of a previously correct system. All these types of event have caused real system breakdowns.

The most common defence against design error is diversity, the use of two or more different and separately designed channels which will be assumed to fail independently. This is expensive, and it still does not protect against errors in the original specification. True diverse implementation of software is surprisingly difficult: even when the specification of a program has been cleanly separated from its implementation, design decisions usually "leak" from the specification into some or all of the implementations.

At some point, all redundant systems need to decide which (if any) channel is faulty. This decision is critical to the operation of the whole system, and some way must be found of making it in a trustworthy fashion.

A simple voter, which merely compares a few logical signals or takes a mean of 3 or 4 analogue values, can be made extremely reliable - the second type (implemented in hydraulic machinery) is used in most aircraft control systems. Digital versions of such a voter are less satisfactory than analogue - deciding whether or not several values are within a reasonable tolerance of one another is much easier in the analogue world! A really simple digital voter cannot tolerate diverse inputs; the channels feeding it must be functionally identical, synchronised, and therefore vulnerable to common-cause failures.

The more diverse the channels of a system are, the more complex the decision maker is likely to be. Obviously it must be substantially more reliable than any single channel, or it would compromise the integrity of the system as a whole. Ideally the voter should be distributed amongst the redundant channels of the system, to minimise the number of critical points at which a single failure would be disabling and to take advantage of diversity in the decision making as well as in the information processing. This leads to "Byzantine voting protocols" and massive overheads, and still leaves some risk of CCF through errors in the specification. Beyond a certain point, complexity may be self-defeating: a more complex system needs better protection simply because it will suffer more frequent failures of individual components.

## AVOIDING COMMON-CAUSE FAILURES

Software does not wear out, and is not susceptible to random failures: all failures of software are the result of design, implementation, or specification errors. In principle a program can be totally correct, and in simple cases formal specification techniques and mathematical verification can be used to show that this is so. Even where full mathematical verification is impractical, graph-theoretic analysis of software is a highly effective method for finding errors and demonstrating that a program meets its specification (Carré (1)). Several toolsets are available to aid this analysis and extend its applicability.

In principle CCF is preventable for software, even if in practice this remains a goal rather than a reality. Diverse implementation of software is therefore a stopgap while more rigorous techniques are developed.

Failure of hardware in contrast can only be postponed, and its effects minimised. Redundancy, therefore, will always be needed in critical systems, bringing with it the risk of CCF. Calculations of mean-time-between-failures are often made with an assumption of independence which may not be justified: if one channel of a system has a measured performance of one failure in 1000 hours operation, it is easy to assume that a 2 channel system will achieve one failure per million hours. No one can measure a reliability of this order, unless thousands of systems are in use: as an example the human body (of which billions are in use) has an MTBF of about 700000 hours.

Figure 1 shows a simple taxonomy of multi-channel redundant computer systems, divided according to the complexity of the processing channels and the voters. Systems of type 3 (simple processors, distributed voting) have too low a performance to be much use, and type 4 (complex processors, simple voters) are very vulnerable to CCF in the

processors which must be functionally identical if the simple voters are to cope. Otherwise each architecture has its strengths and weaknesses:

1 Low cost, low complexity, medium performance, voters simple enough to be made internally redundant and highly trustworthy. Processors must be functionally identical and therefore vulnerable to CCF: processor correctness is critical.

2 Allows use of diverse processors but only at the cost of less reliable voters and increased software cost because of multiple processor types. Voter reliability is critical.

5 The highest cost architecture, with performance to match. Voter reliability is critical.

6 Distributed voting using "Byzantine protocols". Very high reliability at a cost in software complexity and low efficiency. Software correctness is critical.

Architecture 1 has significant advantages for low-cost or lightweight systems, provided its vulnerability to CCF can be overcome.

The specification and design of a microprocessor chip is a relatively simple task by comparison with most software. Like software, the design documents and wiring lists do not wear out and can in principle be totally correct; the chips themselves are mortal like all hardware but actual manufacture is only a small part of the task of producing a microprocessor. It is natural therefore to ask whether the techniques of formal specification and mathematical verification mentioned above in a software context might not be applied to hardware. If these techniques could be used to show convincingly that a microprocessor chip design was totally correct, the risk of common-cause failure would be largely avoided. With this assurance, architecture 1 can be used to build simple, low-cost systems which (with verified software) can give exceedingly high integrity.

**THE 'VIPER' MICROPROCESSOR**

VIPER is a 32 bit reduced instruction set microprocessor which has been specified, designed, and verified using the most formal techniques available. A complete chain of proof exists between the various gate-level designs and the functional specification (Cullyer & Pygott (2), Pygott (3)). VIPERs are designed to work in pairs, using "Architecture 1" above to form fault-detecting computing modules with virtually 100% cover against single faults. All the comparison logic needed is built-in to the VIPER chips and is implemented in duplicated self-checking circuitry to minimise the risk that a single fault in the "voter" might mask faults elsewhere in the system. Every node in the voting system can be tested by applying a few carefully chosen inputs: 4 legal patterns and one deliberately-forced error are enough to test the whole of the 32 bit data bus comparator (Halbert (4)).

With a common specification against which the chip designs have been verified in the formal, mathematical sense, a pair of VIPER chips has the property of dependable fault reporting and forms an ideal building block for reliable systems. Multiple pairs can be used if true fault-tolerance is essential, but majority voting is never needed since a faulty processor pair can be depended on to stop and report its condition.

A pair of VIPERs has exactly the processing power of one VIPER. The fault reporting property does not depend on software so there is no software overhead. In comparison, a system using distributed voting may require as many as 6 processors to deliver 25% of the useful power of one processor alone.

**VIPER Design and Verification**

VIPER was designed almost 5 years ago when even hardware description languages were fairly new. Figure 2 shows the design and verification process in schematic form, from the Top Level Specification (TLS) down to the pattern generator tapes used for manufacture. The TLS is the formal definition of VIPER, used by system designers, compiler writers, and programmers as a reference document. It is only 6 pages long, and is expressed in the Higher Order Logic formalism (HOL) developed at Cambridge University (Camilleri, Gordon, and Melham (5)). An informal English definition of 3 pages exists, but this is for introductory use only.

Details of the verification process have already been published in (2) and (3); now (two years later) it is appropriate to consider what lessons have been learned.

The first lesson is that specifying and verifying a system using formal mathematical techniques is just as difficult and as labour-intensive as the traditional method. Except for very repetitive array structured devices, there is no dramatic time saving at the design stage to be had from the use of formal methods. The savings come later when the device has been fabricated, and when it is being used in a system: there is already ample evidence that formal methods substantially reduce the effort devoted to testing and re-work. Of the four families of VIPER devices made so far, three were logically perfect first time; the first devices produced exposed a fault in the CAD system used for gate-level design and had to be re-worked once.

The second lesson is that, though verification is (in the present state of the art) guided by human mathematicians and therefore fallible, the errors made in a proof and the errors made in a chip design are different in character. Proof techniques are static and declarative ("this relationship holds under the following conditions") where conventional design and simulation are dynamic and operational ("if I do this the following will happen"). It is not much easier to get a specification or proof right, using HOL or any other formalism, than it is to get a chip design or program right, but the errors which people commonly make in the two processes are different. Once the two have been shown to match, there is a high probability that both are correct.

Lesson three is that formal methods take time, and they demand skills which are not familiar to many engineers. The tools available to support them are at present quite difficult to use, having grown out of an academic environment rather than an industrial one. Documentation may be sparse: some techniques can still be learned only by sitting at the feet of the Master.

Some commonplace hardware techniques are difficult to describe formally. Asynchronous logic in general is harder to handle than synchronous, while dynamic logic is harder still. A device built to a formal specification at the moment is likely to be static, synchronous, rather conservative in its use of silicon, and only moderately fast. These limitations match rather closely those which are considered desirable for reliability and testability, so the lesson is that these preferences are well founded. It is rarely wise, in any case, to use the most novel technology in a system with safety or security implications.

## THE LIMITS OF VERIFICATION

Figure 2 and References (2) and (3) summarise the techniques which have been used to verify the design of VIPER. Each level of verification has been approached in at least two ways.

The lower levels are simple but massively tedious; they are done entirely inside a mainframe computer. We have no significant doubt of their correctness.

The two higher levels are mathematically much harder, since the level of abstraction is so much higher: the top level specification knows nothing about electronics let alone silicon technology. The final HOL proof was done in a single step, from TLS to Block Model, and is complete apart from 12 theorems which could not be handled by the HOL prover (Cohn (6)). All have been proved to Cohn's satisfaction by hand. Interestingly, these "difficult" theorems all relate to the well-understood (?) problems of twos-complement arithmetic and overflow.

### Who Proves the Prover?

"Proof" is a typical Humpty Dumpty word, which means precisely what you want it to mean. In practice proof seems to mean "an argument that most practitioners in the field accept as valid", no more and no less. The object of "proving" computer hardware (or software) is to show by analytical techniques that you have actually built what you intended, and that this is so for every possible case. The conventional method of testing can show only that the tests you have actually applied behave as expected; the number of possible tests of any practical computer is astronomically large.

Most of the software tools used to aid complex proofs are far too complex to be proved themselves, though the final proof checker (on which the integrity of the whole process depends) may be an exception. The ultimate check on a proof is to do it again using different methods, and this we have tried to do by simulation and by repeating the top level proof: once with pencil and paper and later with the HOL Theorem Prover. The aim as always is to minimise the risk of CCF in the eventual system, in this case using diversity of proof techniques instead of diverse chip designs.

### The Lower Limit of Proof

The formal verification of VIPER extends from the Top Level Specification to the "wiring list" which forms the input to conventional CAD tools.

However, the wiring list is not a chip. Several layers of CAD software, mask making, photolithography, ion implantation and so on interpose between the two. Checking the pattern-generator tapes against the wiring list is a straightforward one-to-one comparison, but at the level of basic physics formal logic is not very useful: physics is not constrained to obey the rules! Verification of the design gives a strong assurance that two identical VIPERs will not fail simultaneously because of some inbuilt design fault, but there is always the risk of an obscure pattern-sensitive fault (e.g capacitive coupling between conductors on the chip) which is common to all VIPERs in a given technology but cannot be exposed by any reasonable amount of testing.

At this level diversity is the only defence, but fortunately it is not difficult to exploit. Independent implementation of two chips, to a common logical and timing specification, is a much easier problem than independent production of software since most of the work is done by the (different) CAD tools. The starting point for each implementation is the Block Model (Figure 2), which is common to all and assumes the existence of relatively abstract structures like adders, registers, and multiplexors. Transformation of these into networks of gates and conductors can be done fairly

quickly; in the latest implementation the transformation was almost wholly automatic. Once generated, a gate-level implementation can now be verified in a few hours using the NODEN system (3).

**BUILDING A DEPENDABLE SYSTEM**

For many applications, a single pair of VIPERs is adequate. Dependable fault reporting (followed by shut-down or reversion to a safe state) is just what is needed for most plant monitoring systems, electronic funds transfer, or medical instrumentation. For continuous control, 2 or 3 pairs must be used with independent power supplies and clocks for maximum integrity; all pairs take inputs but only the working pair generates outputs. In every case the ability of a VIPER pair to stop and report any error is critical, so an analysis of fault coverage is necessary.

Figure 3 shows a typical VIPER pair in schematic form, with (X) showing the locations of typical faults either in the chips themselves or in the interconnexion. Interconnexion faults are in practice much the more common! In normal operation one VIPER is "active" and drives the various data, address, and control signals, while the other is the "monitor" and receives the signals on corresponding pins to be checked against its internal equivalents. Any difference causes the STOP signal to be asserted, which then stops the active processor as well. The comparison logic is active in both processors, so that a short-circuited bus line would be detected by both. The "Major State" bus operates the other way round, and is driven by the monitor (see Categories 4 and 5 below).

Faults are not restricted to any particular model: any event which leads to a wrong signal will be detected provided it does not afflict both VIPERs identically. Deliberate fault injection is necessary at intervals to check that the error reporting system is working, but because of the self-checking design of the comparison logic only one bit of each bus needs to be "flipped" for a complete check. The check generates a sub-microsecond pulse on the STOP lines which can be used to confirm that it has been performed. The box marked "inject" contains 5 exclusive-OR gates: for a full check two of the control lines need to be flipped.

Nearly all faults fall into one or other of the following categories:

1 Faults in the internal workings of either VIPER, in the data bus, address bus, or control signals between the two VIPERs: the monitor VIPER compares the signals with its internal values at every memory cycle, and asserts STOP if they do not match.

2 Faults in the memory, or in the "spur" connecting each memory chip: 8 parity bits are appended to the 32 bit data bus, giving 100% detection of multiple errors confined to one 8 bit byte. 96% of random multiple errors are also detected, but the emphasis is on failures of a single chip. Coverage depends on each memory chip having its own spur from the bus, otherwise a single fault might affect more than one byte of memory. Parity errors cause both VIPERs to assert their STOP signals.

3 Faults in decoders which select particular groups of memory or I/O chips: the parity scheme depends on each byte of the memory being self-contained, with its own address decoder, so for total coverage each byte-wide slice of the memory system should have its own decoder chip - 5 decoders including the parity byte. In practice very good coverage is obtained with 2 decoders, driving 3 bytes and 2 bytes. Many VIPER systems need only two sets of memory chips (RAM + PROM) and the decoder is no more than an inverter.

4 Faults in the "Major State" bus between the VIPERs: the Active VIPER detects these and stops. The reversal of direction on this bus provides a back-up means of stopping the processors in the event of ...

5 ... a fault in one of the "STOP" lines linking the two VIPERs, which could conceal a second fault. A "stopped" VIPER enters a unique Major State and stays there.

6 Faults in the I/O interface chips are covered in the same way as memory faults. Faults in the sensors, actuators, or wiring are detected by "tellback" signals in the usual way, preferably using different I/O interface devices and/or different positions in the 32 bit data word for outgoing and return signals.

7 Faults in the clock generator. These are serious only if they affect both VIPERs, since a unilateral fault will cause the buses to mismatch almost immediately. However, a pair of VIPERs must be clocked in exact synchronism so most systems will use only one clock oscillator per pair. Clock failure is best detected by simple pump-up timers, which can be duplicated and refreshed either directly from the clock signal or by outputs from the software. A software-driven timer has the advantage of protecting against a much wider range of faults.

8 A few pathological faults inside a VIPER chip which can in theory mask subsequent faults. Nearly all of these are found by fault injection (which then fails to provoke an error) but this may be possible only at long(ish) intervals during normal operation.

9 Failure of the mechanism which reports failures, shown as a simple OR gate in Figure 3. Obviously this must be duplicated, preferably using complementary logic.

**'VIPER' IN INDUSTRIAL SYSTEMS**

Everyone expects the best available technology to be used in any system which could pose a threat to life or the environment. At the moment the best available technology in computer control systems involves diverse implementation of both hardware and software, complex distributed voting protocols, and massive processing power to offset the overheads. Systems with several hundred microprocessor chips have been proposed. The cost and size of such systems rules them out for all but a few applications.

Everyone also makes mistakes from time to time. Occasionally a human mistake has serious consequences, which could have been prevented by a trustworthy automatic system. There are many areas of life, from the hospital to the kitchen, which could be made safer if the cost and size of ultra-reliable computers could be reduced. Though VIPER began as a test-vehicle for formal methods of hardware design, and was first realised as a "rugged" chip for specialised military applications, its real significance is the hope it offers for genuinely low-cost systems of very high integrity.

Candidate applications are everywhere:

a Factory automation, which injures many and kills a few every year.

b Medical electronics. Only one patient may be at risk per system, but the potential number of (say) heart pacemakers is large.

c Automotive systems: engine management, anti-lock braking, active suspension, transmission control.

d Mass transit systems, which may be required to operate round the clock for social reasons but are difficult to staff at night.

e Domestic equipment such as microwave ovens and washing machines, which injure an alarming number of children every year.

f Aids to the disabled, which might leave the user helpless and possibly in danger if they fail.

g Fire alarms, security and surveillance equipemnt. False alarms are economically as serious as failure to alarm.

h Robots for factory, home, or garden ... and so on, and on.

The Choice of Configuration

The most obvious way of constructing a fault-reporting VIPER system is to put two VIPER processors on the same Printed Circuit Board. This is not always the best configuration: full fault-coverage requires that all the memory and peripheral interfaces be placed electrically "between" the two VIPERs (Figure 3) and this may not be possible in a small space. For a compact system though, with a single well-defined interface (RS 485 or Mil 1553b for instance) a self-contained fault reporting module on a PCB or hybrid substrate may be the answer.

For low cost applications, or prototype construction, a bus-organised system is convenient. Full fault coverage is difficult with a conventional bus, though it is possible with a "daisy chain" arrangement where each signal enters and leaves the card on separate pins: this technique is used in the current VIPER Prototyping System. On an industry standard bus like STE, quite good coverage can be achieved by putting the two VIPER processors on adjacent cards with a front connector to link those signals which are not defined on the bus: standard techniques like duplication of outputs and "tellback" can then be used to give very high integrity at the system level.

A third possibility is to use the VIPER design methods rather than the chip itself. Many simple applications do not need the power of a microprocessor, and for these an ASIC is often the best solution. Formal methods of specification and verification have been proved to work for static, synchronous devices like VIPER, though in principle they can be applied to any deterministic circuit. Recently, field programmable gate arrays (FPGA) have become available which have a pre-defined clock distribution system built in to the substrate; these match the VIPER design philosophy closely and could probably be used to re-implement a simplified version of VIPER. The verification process would contain a "weak link" where the formal language (e.g HOL) was translated into the appropriate FPGA design language, but our experience with HOL and ELLA suggests that this process is easy enough to be trustworthy. However, ELLA is itself a formal language with a mathematical basis very close to that of HOL (Morison et al (7)); other languages could be more difficult. Simulation on either side of the "weak link" can always be used to build up confidence.

In the long run, as chips and systems become more complex, mathematical techniques of design and verification will become the only practical way of building correct systems. Simulation of a complex device can explore only a tiny part of its possible behaviour; as an example the VIPER chip has more than 2 to the power 200 internal states

and no simulation could possibly cover all of them. In contrast, algebraic verification by its nature covers all states of a system, and beyond a certain level of complexity it will be the only truly dependable method.

**REFERENCES**


1 Carré B. A. 1979, "Graphs and Networks", Oxford University Press.

2 Cullyer W. J. and Pygott C. H. 1987, "Application of Formal Methods to the VIPER Microprocessor", Proc.IEE, 134, 133-141.

3 Pygott C. H. 1988, "NODEN: An Engineering Approach to Hardware Verification", Proc. Workshop on the fusion of hardware design and verification, ed. Milne. North Holland.

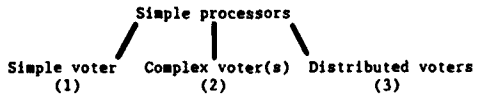
4 Halbert M. P. 1987, "A self-checking computer module based on the VIPER microprocessor", Proc. Safety & Reliability Society Symposium, Altrincham, UK.

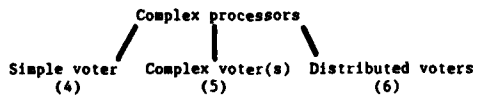
5 Camilleri A., Gordon M., and Melham T. 1986, "Hardware Verification using Higher Order Logic", Proc. IFIP International WG10.2 Working Conference, North Holland.

6 Cohn A. 1987, "A Proof of Correctness of the VIPER Microprocessor: the First Level", Proc. Workshop on the Verification of Hardware, Calgary, Canada. Kluwer Academic Publishers 1988.

7 Morison J. D., Peeling N. E. and Thorp T. L. 1985, "The design rationale of ELLA, a hardware design and description language", Proc. Conference on Hardware Description Languages and their applications, Tokyo, Japan.




| | (1) | (2) | (3) | (4) | (5) | (6) |
|---|---|---|---|---|---|---|
| Performance | Medium | M | L | H | H | M |
| Complexity | Low | M | L | M | H | M |
| CCF risk | High | L | L | H | L | L |
| Cost | Low | M | L | M | H | M |
| Voter reliability | High | L | H | H | L | H |
| Software cost | Low | M | H | L | M | H |

**Figure 1 Taxonomy of redundant systems**

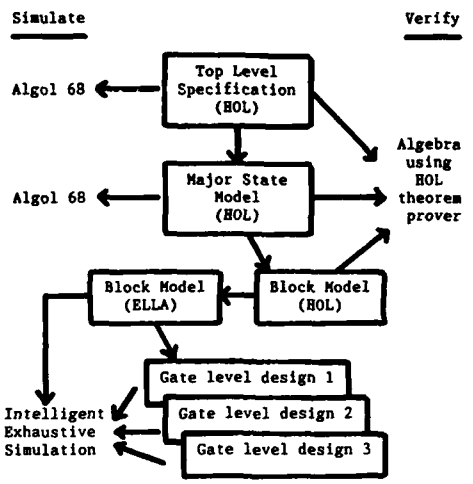



Figure 2 The VIPER Design Process

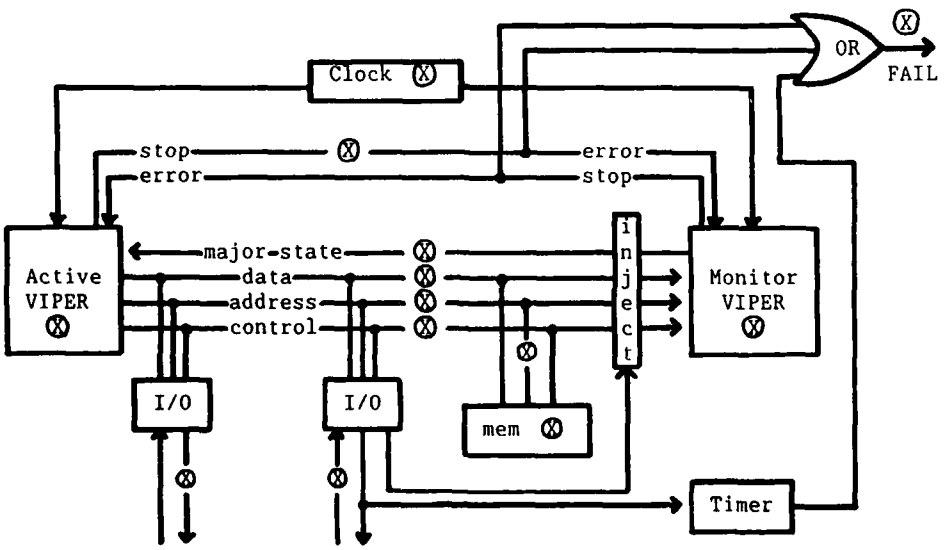



Figure 3 Fault-Reporting Computing Module

# FAULT-TOLERANT, FLIGHT-CRITICAL CONTROL SYSTEMS

by

Tom Sadeghi, Manager
Advance Flight Control Engineering

Gerry Mayville, Program Manager
Self-Repairing Flight Control Systems

General Electric Company
Aircraft Control Systems Department
P.O. Box 5000
Binghamton, New York 13902, USA

## 1. ABSTRACT

This paper describes and compares two leading Fault-Tolerant, Flight-Critical Systems (FTFCSs) currently being developed at General Electric (GE) Aircraft Control Systems Department (ACSD). These technologies, driven by the aircraft performance, reliability, and maintainability requirements, are: the Self-Repairing Flight Control System (SRFCS) and the Vehicle Management System (VMS). SRFCS has two technology thrusts: Control Reconfiguration Strategy (CRS) and Onboard Expert System (OES). VMS is focused on: Vehicle Management Computer (VMC) development and Integrated Diagnostics System (IDS). SRFCS has the potential to reduce brute force hardware redundancy, where the VMS is driven by increased functional complexity demands for increased hardware redundancy. A cursory examination of these technologies suggests that SRFCS can be considered as a complement to VMS development. Contrary to this view, the paper examines the attributes of each of these technologies and identifies the needs for future development. The remaining challenge to be overcome by systems designers is finding the best balanced solution for the future FTFCS, utilizing a proper blend of SRFCS and VMS technologies.

## 2. INTRODUCTION

The Fault-Tolerant, Flight-Critical Systems (FTFCSs) must comply with the requirement of continual safe operation of an aircraft after loss of two functional paths within the flight critical system. This requirement is translated into the design of systems with adequate redundancy to comply with the fail-operative/fail-operative design criterion. This level of redundancy is currently achieved through implementation of either a triplex or a quadruplex flight control system, i.e., brute force hardware redundancy. Other requirements considered as drivers for designing FTFCSs include:

- Probability of Mission Abort (PMA) $\leq 10^{-3}$ per flight hour
- Probability of Loss of Control (PLOC) $\leq 10^{-5}$
- Probability of detecting and isolating a critical failure mode to a functional path > %95

These requirements are achievable through the use of hardware redundancy with deterministic operation to satisfy the safety-of-flight requirements.

The SRFCS technologies are focused on exploiting the inherent redundancy among the control surfaces of an aircraft to satisfy the FTFCS requirements, and:

- To reduce hardware by degrading hardware redundancy, thus improving reliability
- To provide the same level fault tolerance as in current systems, thus satisfying mission and safety requirements
- To increase fault isolation coverage, thus improving maintainability
- To reconfigure control laws to compensate for battle damage surfaces, thus improving survivability

The SRFCS technologies have been developed over a decade under funding from Air Force Flight Dynamics Laboratories (FIGL and FIGX).

One of the SRFCS technology thrusts has been to develop Control Reconfiguration Strategies (CRSs) capable of redistributing nominal flight control commands to alternative surfaces to compensate for a battle damaged surface.

Another technology thrust for SRFCS has been the development of improved diagnostics for the flight control systems. The diagnostics objectives are to provide capabilities for detecting, isolating, and reporting failures that are, otherwise, categorized as Cannot Duplicate (CND) and Retest O.K. (RTOK). Artificial intelligence-based expert system technology has been used to achieve the flight control system diagnostic objectives.

GE/ACSD, under contract to Air Force and MCAIR, has been developing the CRS and OES that will be flight tested by MCAIR on F-15 HIDEC during 1989.

## 3. SELF-REPAIRING FLIGHT CONTROL SYSTEMS (SRFCSs)

SRFCS technologies developed by GE/ACSD are in two parts: Control Reconfiguration Strategy (CRS), and the Onboard Expert System (OES). Each of these technologies will be described in the following text.

### 3.1 Control Reconfiguration Strategy (CRS)

CRS performs three functions: control power distributions following battle damage or an actuator failure (Figure 1), surface damage or actuator failure detection and isolation (Figure 2), and battle damage estimation (Figure 3).

These functions are implemented downstream for the nominal flight control system and upstream for the actuators as shown in Figure 4. The constituents of the CRS are:

- ***System Detection, Isolation, and Classification* (SIDC).** SIDC uses *aircraft sensor data, actuator commands, and* actuator sensor information to detect and isolate a damaged surface or a failed actuator. This function is performed by using the hypothesis testing techniques and Kalman filter developed by Alphatec Company under contract to GE/ACSD.
- **Effector Gain Estimation (EGE).** EGE uses an extended Kalman filter and aircraft dynamic models (normal and impaired) to estimate the effectiveness of a partially missing surface.
- **MIXER.** This CRS component uses a pseudo inverse of control derivative matrix to redistribute the control signals generated by the flight control computer to the remaining surfaces to compensate for the surface loss or actuator failure.

Figures 5, 6, and 7 represent the pitch, yaw, and normal acceleration responses of the F-15 HIDEC under normal conditions, 100% loss of right stabilizer without reconfiguration, and after reconfiguration. These responses were obtained by simulating the failure on a nonreal-time, nonlinear simulation of the F-15 HIDEC aircraft at GE.

As shown by these figures, CRS has the capability of compensating for battle damage and actuator failures by reconfiguring the flight control laws. This capability can be exploited to reduce the brute force hardware redundancy currently used to implement flight control systems. However, the limitations of the current CRS techniques remain to be resolved. These limitations include:

- The need for impaired models of the aircraft resident in the Flight Control Computers (FCCs) to compute the reconfiguration gains
- Sensitivity of the fault detection schemes to sensor noise
- Flutter problems associated with damaged surface or damaged aircraft

Alternative potential technologies to overcome the limitations of the current CRS techniques are:

- Model reference adaptive control techniques to eliminate the need for storing impaired models of the airplane in FCCs
- Electrohydrostatic Actuation (EHA) technology with electronic stiffness capabilities to solve the flutter problems
- Robust failure detection schemes, using banks of Kalman filters

If CRS were to be used in a flight control system to degrade hardware redundancy, then it would require fault detection schemes that are more reliable and have the capability of isolating failures to Line Replaceable Units (LRUs) with a much higher level of confidence than exists currently in flight control systems.

The enhanced diagnostics capabilities are also needed to improve the maintainability parameters of the flight control systems. A leading technology that has shown many promises to solve the maintainability problems is the Onboard Expert System (OES).

### 3.2 Onboard Expert System (OES) for Maintenance Diagnostics

GE has been developing and applying OES technologies to avionics systems for more than a decade. The GE components involved in the development and application of expert systems to avionics include: GE Corporate Research and Development (CR&D), GE Automated Systems Department (ASD), and GE Aircraft Control Systems Department (ACSD). Figure 8 represents the evolution of expert system technologies at GE. On the left-hand side of the figure, the generic diagnostic problems are shown graphically. These problems consisted of isolating faults to the Line Replaceable Unit (LRU), to the functional level, and to the circuit level.

The technologies applied by GE to solve the diagnostics problems are listed here in chronological order:

- **Test Engineers** using test equipment to troubleshoot failed components
- **Rule-Based Reasoning Expert System (GEN-X)**, developed and applied by GE/CR&D
- **Onboard Expert System (OES)**, developed and applied by GE/ACSD
- **Framed-Based Expert System (ALBERT)**, developed and applied by GE/ASD
- **Reasoning with Uncertainty Modeling (RUM)**, with fuzzy logic reasoning capability, developed and applied by GE/CR&D
- **Model-Based Reasoning (MBR)**, under development at GE/CR&D, which combines the features from frame-based and RUM expert systems

These techniques have been used in Automatic Test Equipment (ATE), and flight and engine control maintenance diagnostics.

The OES developed by GE/ACSD was first conceived under a joint GE Research and Development (R&D) fund and a contract from Air Force Wright Aeronautical Laboratories (AFWAL) in the mid-1980s. The development evolved over a period of years, and the most current activity has been the F-15 HIDEC application under a contract to MCAIR, and under the sponsorship of NASA Dryden and AFWAL-FIGX. The Phase 0 of the OES has been successfully flown by MCAIR on F-15 HIDEC during the first quarter of 1989.

The objective of Phase 0 was to demonstrate the capability of detecting and isolating an Angle-of-Attack (AOA) failure mode. This failure mode is intermittent, appearing during aircraft maneuver by having, for example, connector pins disconnected under *g* stress. The flight control system detects the failure, declaring an AOA failure. The aircraft requires maintenance action. In this case, after touch-down, the maintenance crew removes the flight control computers' LRUs. Examination of the FCCs by the technicians would reveal no failure in the FCCs; therefore, the failure cannot be duplicated (CND condition).

The OES has been developed with the capability of detecting and isolating the failures in flight by using expert system inference techniques. The attributes of OES and the AOA failure scenario are shown in Figure 9.

Phase 1 of the OES program consisted of demonstrating six failure scenarios that were representative of real life situations and were of types that would generate unnecessary maintenance actions. The top-level OES structure is shown in Figure 10. The OES communicates with the flight control system's 1553 bus, the CRS, and a tape drive through its own real-time Executive. The inference engine represents the brain of OES, with the capability of forward and backward chaining information logic to interrogate the health of the system and to detect and isolate failures.

OES interrogates the input signals from the flight control systems for detecting and isolating a failure. The inference engine employs rule tables. Rule tables represent logical relationships among the subsystems and signals. The signal-to-symbol converter module translates raw signals to true or false logic. To demonstrate OES capabilities in flight and without physically impairing the aircraft, it was necessary to emulate the failure modes. The impairment emulation models were built and incorporated inside the OES for each failure scenario as shown in Figure 10.

The failure scenarios developed for demonstration during Phase 1 of the program are shown in Figure 11. The maneuver conditions, the system failure indication, the pilot reaction in response to the failures, the declaration of failed subsystem, and the actual cause of the failure are shown and categorized in the figure. For these failure scenarios, the OES would identify the actual cause of the failure and the actual failed subsystem.

This expert system maintenance diagnostics technology will be flight tested during the last quarter of 1989 by MCAIR under sponsorship of NASA Dryden and AFWAL-FIGX.

## 4. HARDWARE REDUNDANCY AND DIAGNOSTICS

Current flight control systems use hardware redundancy to achieve the fault tolerance requirements. Next-generation flight-critical systems must provide more intelligence and control for aircraft subsystems, must improve functional capabilities, thus increasing functional complexity, and must integrate the electronics used to implement these functions. These requirements define the concept of a Vehicle Management System (VMS) replacing the traditional flight control systems. The VMS requires fault detection and isolation capabilities to Line Replaceable Module (LRM) with greater than, or equal to, 99% coverage. To reduce the burden on the logistics support, common module electronics are required within the aircraft subsystem avionics as well as across different types of flying vehicles. The vehicle management system requires improved reliability, maintainability, and supportability over the predecessor flight-critical systems.

### 4.1 Vehicle Management System (VMS)

The requirements for Fault-Tolerant, Flight-Critical Systems (FTFCSs) suggest that channelized architectures with triplex or quadruplex redundancy are preferred solutions for developing vehicle management systems. The VMS requires a deterministic operation to guarantee safety of flight. Advanced architectures with dynamic reconfiguration of hardware and/or software face great difficulty in satisfying the safety-of-flight requirements, and lack sufficient verification and validation means. They require a tedious and complex hardware/software integration, and testing processes. Formal proof of the concept, therefore, is hard to achieve.

A VMS must be capable of performing identical redundant computations to demonstrate a verifiable operation with no performance degradation in the presence of two like failures.

When designed correctly, channelized architectures, operating synchronously, are capable of satisfying the minimum transport delay requirements in closing aircraft and actuator control loops. Figure 12 represents a qualitative comparison of a leading quadruplex architecture with a triplex architecture, with each channel having a Self-Checking Pair (SCP) of central processing units. The overall conclusion drawn from this figure suggests the preference for quadruplex architectures, despite having more parts and failure rates over triplex architectures with self-checking pair processors.

### 4.2 Integrated Diagnostics System (IDS)

A fault-tolerant, flight-critical system architecture for VMS must comply with the maintainability requirements of two-level maintenance. This requirement translates into a VMS definition that has high fault isolation coverage with a high level of confidence, isolating faults to Line Replaceable Module (LRM) level.

Another requirement for developing a VMS architecture is the use of Common Modules (CMs), with commonality within aircraft subsystems as well as across aircraft types. The common modules must be implemented using SEM-E size standard modules definition. This requirement forces the flight control functions to be partitioned over more boards than are found in traditional flight control computers.

The two-level maintenance requirement, with fault isolation capability to LRM levels, translates into adding many tests and monitoring circuits to each computer board. These requirements increase the redundancy management and Built-in Test (BIT) functions, and the size and complexity of the associated software. Figure 13 summarizes the VMS Maintainability Values (M-Values) versus issues associated with their implementation.

To address the maintainability issues of a vehicle management system design, an Integrated Diagnostics System (IDS) approach has been developed. Figure 14 describes the constituents of IDS and their capabilities.

The IDS is defined by four levels of diagnostics:

**Level 1** relies solely on the fault detection and isolation attained from the Redundancy Management and BIT (RMB) designed into the channelized VMS. For a quadruplex architecture, RMB can provide 98% fault detection, and isolation to a functional path (or LRU for some failure modes).

**Level 2** of diagnostics employs an Onboard Expert System (OES) to provide improved fault detection and isolation coverage by diagnosing faults that cannot be duplicated otherwise on the ground.

**Level 3** of diagnostics is designed for a small set of failure modes that belong to an ambiguity group nondiagnosable in flight. This diagnostic level is designed into a Portable Maintenance Aid (PMA). PMA will guide a maintenance technician to perform additional tests (Initiated BIT, or Loop-on Test) to further isolate an ambiguous fault to an LRM.

For those failures that cannot be isolated to an LRM with a high degree of confidence, **Level 4** of diagnostics will accept the in-flight fault code as well as pilot and maintenance crew comments to further isolate a failure. Level 4 diagnostics has access to a central maintenance computer in order to utilize historical data from other fleets and operations.

Levels 2 through 4 of the diagnostics use various expert system technologies to achieve their intended functions. Figure 15 represents how these levels of diagnostics are utilized during various segments of a mission.

## 5. CONCLUSION

Current Control Reconfiguration Strategies (CRSs) and Onboard Expert System (OES) technologies have potentials to transition into a production airplane, but require improvements and flight tests before qualification. New hardware architectures for Vehicle Management Systems (VMSs) and Integrated Diagnostics Systems (IDSs) have many promises worth pursuing their developments, but have a long way to reach the full proof of the concept in order to transition into a production aircraft. CRS/OES technologies as well as VMS/IDS technologies face many technical challenges ahead.

The idea of combining or integrating these technologies into one platform is, naturally, a topic belonging to the future.

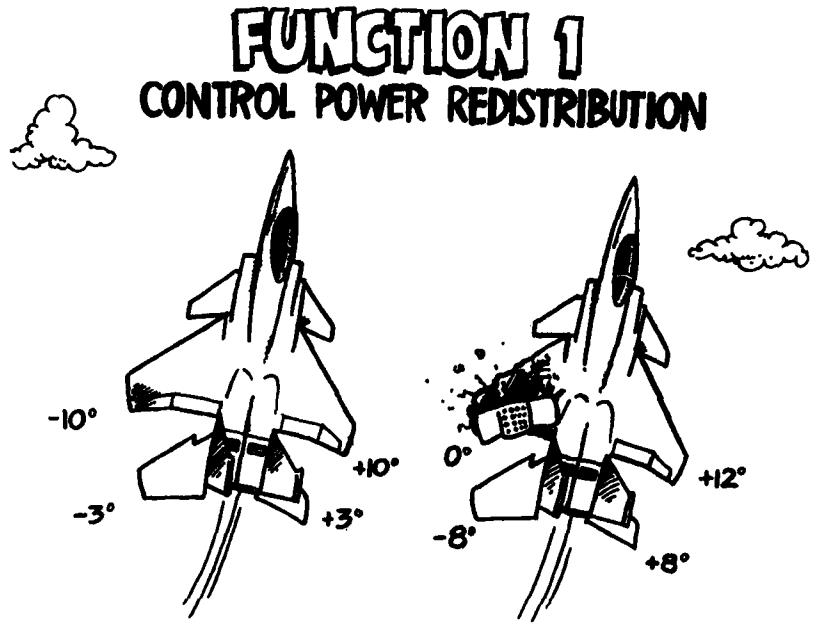



Figure 1. Control Power Redistribution

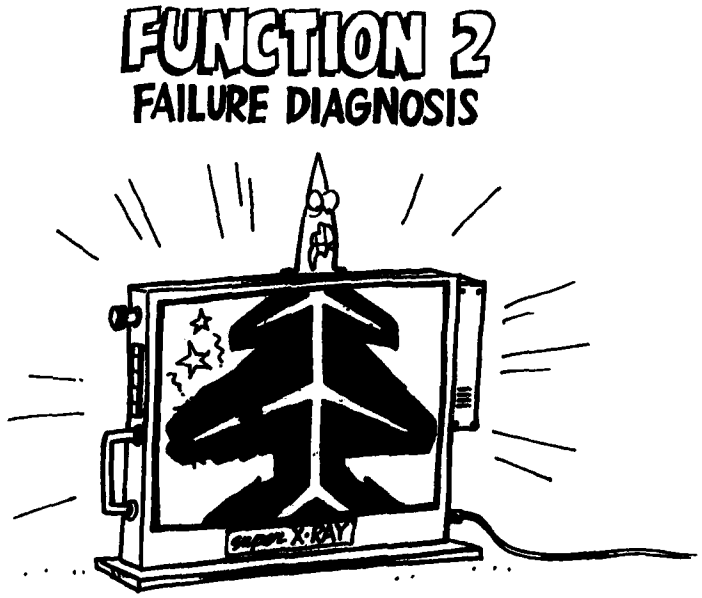



Figure 2. Failure Diagnosis

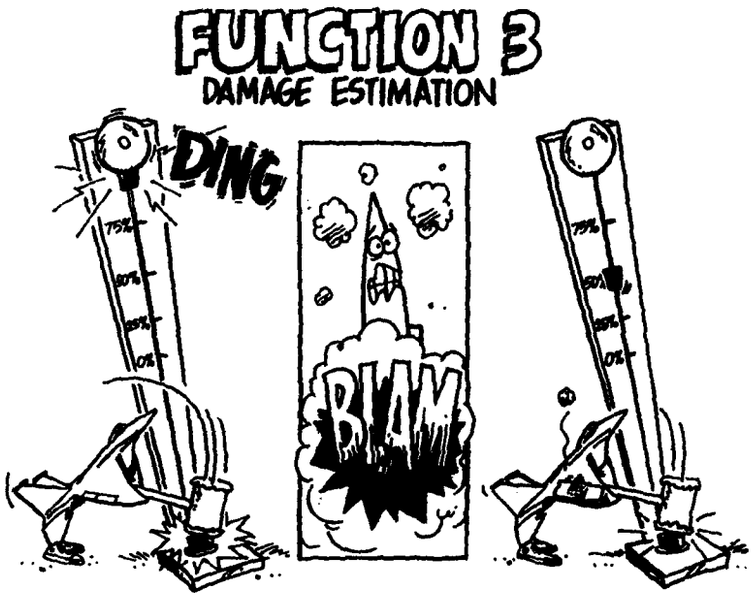



Figure 3. Damage Estimation

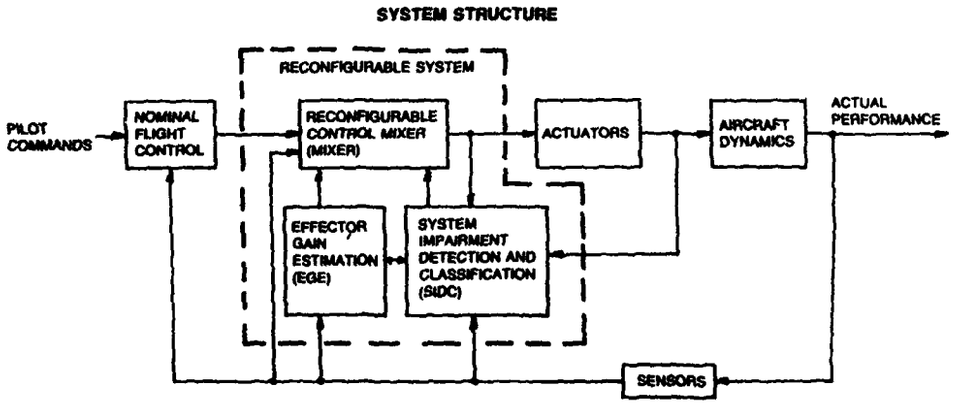



Figure 4. Reconfigurable Flight Control System

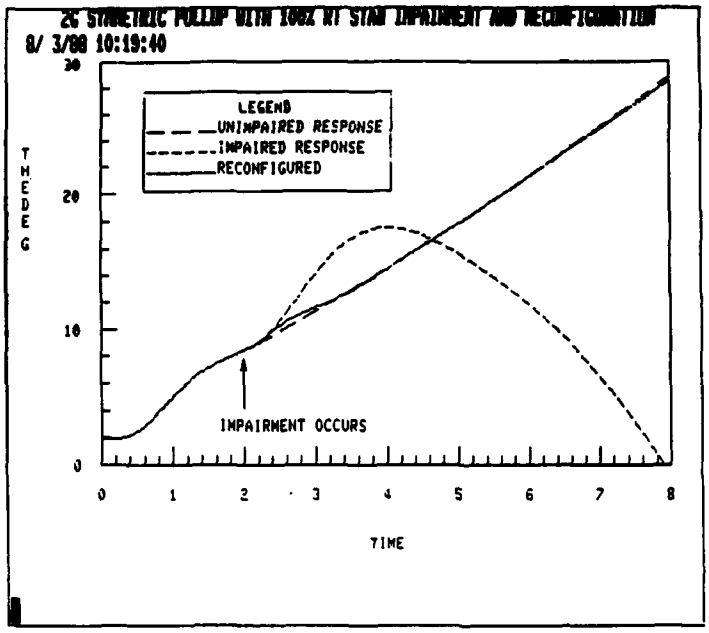



**Figure 5. 2G Symmetric Pullup with 100% Rt Stab Impairment and Reconfiguration**

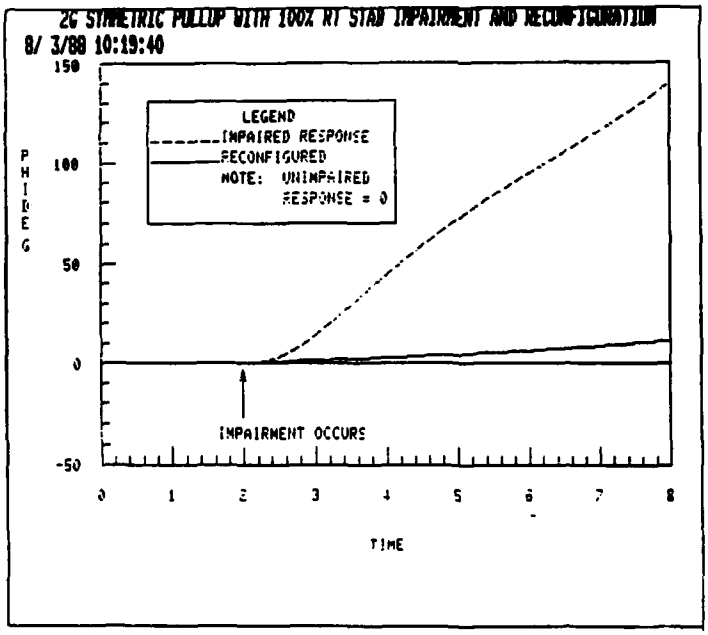



**Figure 6. 2G Symmetric Pullup with 100% Rt Stab Impairment and Reconfiguration**

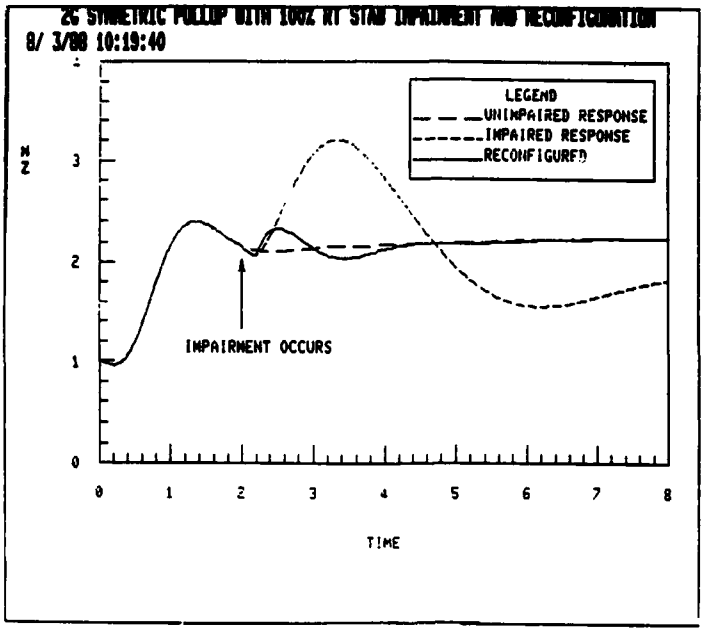



Figure 7. 2G Symmetric Pullup with 100% Rt Stab Impairment and Reconfiguration

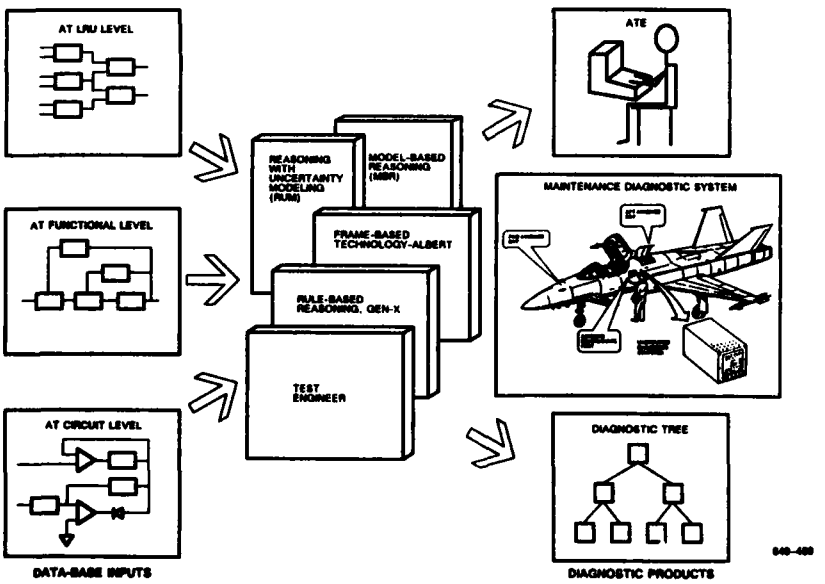
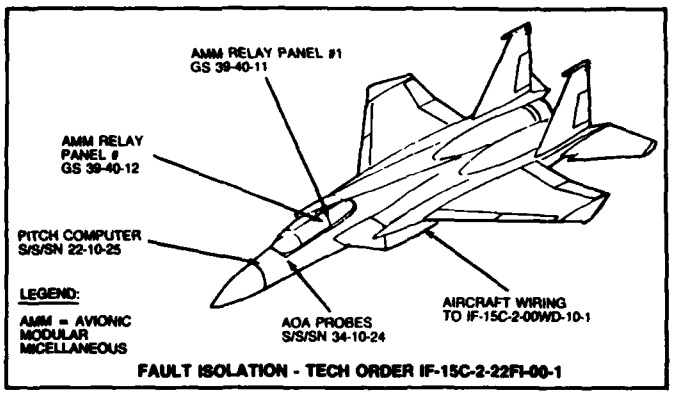


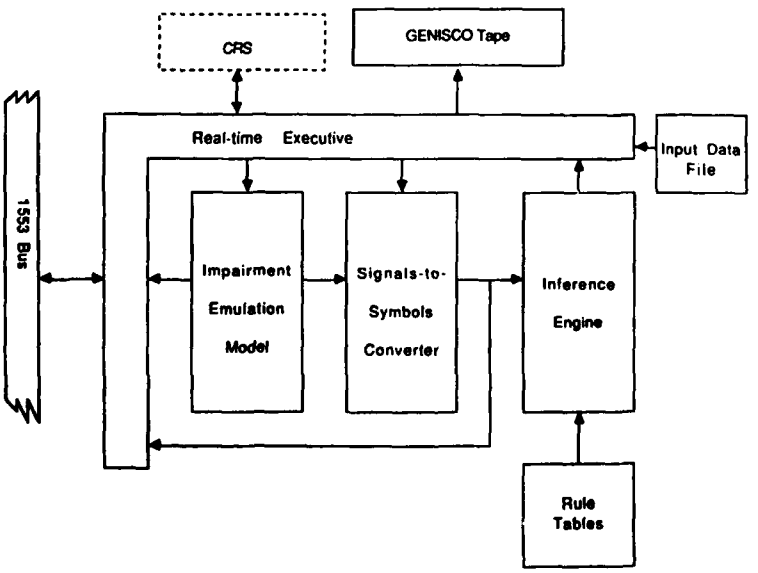

Figure 8. Diagnostic Technology Evolution at General Electric Company




- ANGLE-OF-ATTACK CHANNEL DIAGNOSIS
- IBM PC/GEN-X ARCHITECTURE
- RULE BASED (≈50 RULES)
- ISOLATES TO 1 OF 75 REPLACEABLE ELEMENTS

649-488

**Figure 9. F-15 Demonstration**




649-489

**Figure 10. OES System Structure**

DATE: 8-16-88

| | MANEUVER CONDITIONS | FAILURE INDICATION MAJOR SYSTEM | DISPLAY | PILOT ACTION | SUBSYSTEM FAILED | CAUSE |
|---|---|---|---|---|---|---|
| 1. | 3g LEVEL TURN | ROLL CAS DISENGAGES (EMULATED) | POESI | GO TO SLF AND RESET CAS (EMULATED SW) | DYNAMIC PRESSURE SENSOR | CONNECTOR FAILS UNDER g LOAD |
| 2. | SLF, SMALL PITCH INPUTS | PITCH, ROLL CAS DISENGAGE (EMULATED) | POESI | RESET CAS | STAB ACTUATOR (WITH RECONFIG.) | HYDRAULIC OR ELECTRICAL FAILURE |
| 3. | SLF, SMALL PITCH INPUTS | PITCH, ROLL CAS DISENGAGE | MASTER CAUTION P CAS LIGHT R CAS LIGHT | RESET CAS | STAB ACTUATOR (TRUE FAILURE SIGNAL) | HYDRAULIC OR ELECTRICAL FAILURE |
| 4. | 50 DEG/SEC ROLL RATE | A/P DISENGAGE (EMULATED) | POESI | GO TO SLF | INS | PLATFORM STABILIZATION FAILS AT HIGH ROLL RATE |
| 5. | 3g LEVEL TURN | PITCH, ROLL CAS DISENGAGE (EMULATED) | POESI | GO TO SLF, AND RESET CAS (EMULATED SW) | PITCH COMPUTER | CARD A LOOSE CONNECTION UNDER g LOAD |
| 6. | PITCH STICK INPUT TO CHANGE AOA | PITCH, ROLL CAS DISENGAGE (EMULATED) | POESI | GO TO SLF, AND RESET CAS (EMULATED SW) | AOA PROBE | FRICTION BINDING |

NOTATION: SLF MEANS STRAIGHT AND LEVEL, UNACCELERATED FLIGHT

**Figure 11. OES Phase 1 –Failure Scenarios**

| Discriminators | Quadruplex | Triple-SCP |
|---|---|---|
| Redundancy | 2 Fail-Operate (Fail-Safe) | 2 Fail-Operate |
| FDIA Approach | 1st Flight-Critical Fault: Vote<br>2nd Flight-Critical Like Fault: Vote<br>3rd Flight-Critical Like Fault: Vote<br>To detect: shut down sensor, actuator, or channel for fail-safe; or operate on 4th channel | 1st Flight-Critical fault: Vote<br>2nd Flight-Critical Like Fault: Self-Monitor<br>3rd Flight-Critical Like Fault: Self-Monitor Causing Loss of Control |
| Coverage | Lower than Triple-SCP for First Fault<br>Higher than Triple-SCP for 2nd and 3rd | Higher than Quadruplex for First Fault<br>Lower than Quadruplex for 2nd and 3rd Like Faults |
| | Less Complexity in Achieving Requirements | More Complexity in Achieving Requirements |
| Probability of Mission Abort | After 2 Like Failures (Remaining 2 Channels) or after 3 Like Failures in Some Subsystems | After 2 Like Failures (Remaining 1 Channel) |
| | Higher than Triplex with Higher level of Confidence | Lower than Quadruplex with Lower Level of Confidence |
| Probability of Loss of Control | After 4 Like Failures Will Meet the Requirements with Higher Level of Confidence than Triplex | After 3 Like Failures Will Meet the Requirements with Lower Level of Confidence than Quadruplex |
| MTBF (How Often Fails) | Less Than Triple-SCP—More Parts Count *Not* Sufficiently Low to Impact Operation | Higher than Quadruplex—Fewer Parts Count<br>*Not* Sufficiently Higher To Discriminate |
| Complexity: (HW, SW, System) | Lower than Triple-SCP | Higher than Quadruplex |
| Costs | Lower than Triple-SCP | Higher than Quadruplex |
| Weight | Higher than Triple-SCP | Lower than Quadruplex |
| Failure Transients | Lower than Triple-SCP | Higher than Quadruplex |

**Figure 12. VMS Architecture Discriminators**

- Maintainability Goal: Two-Level Maintenance Operational Scenario
- Means to Achieve the Maintainability Goal: High Fault Isolation Coverage with High Level of Confidence
- *Common Module/Standard SEM-E Size versus Fault Isolation Coverage (* > 95%)/Confidence (* > 95%)*

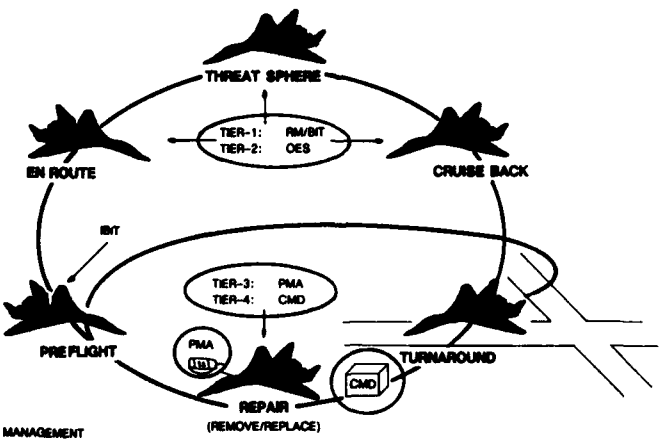
| M-Value | Issues |
|---|---|
| (N/A) | CM/SEM-E will reduce operational spares; thus, logistics support cost. |
| (-) | CM/SEM-E will require flight-critical functions to be distributed over more LRMs/SEM-E boards; thus, fault isolation coverage to achieve the maintainability goal will become more difficult to accomplish. |
| (+) | CM/SEM-E onboard maintenance processor can provide higher fault coverage for onboard circuitry in the absence of failure modes affecting the M-processor. |
| (-) | Failure detection and isolation of the interface media, sensors, actuators, and associated analog circuitry are not improved by having an onboard M-processor. |
| (-) | CM/SEM-E partitioning of, for example, an analog feedback and feedforward for an actuator will require the M-processor to have extra I/O with other boards in order to complete a wraparound or a model comparison test at lower coverage. The M-processor, in this case, has to deal with ambiguity groups associated with the interfaces between boards. |
| (-) | For analog CM/SEM-E boards, one must either implement the A/D and D/A conversions on each board or use excessive dedicated lines that translate into added complexity and failure rates, thus lowering maintainability. |
| (-) | Timing-related faults in either the maintenance or functional processor may not be detected and isolated properly because of dependency between the oscillator and the software-implemented time-out timer. |
| (-) | CM/SEM-E onboard processor functionally acts as a monitoring device, but its failure rate is not less than 1/10 of the failure rate of the functions it is monitoring (as required). |
| (-) | CM/SEM-E onboard processor reduces the board's MTBF, thus degrading maintainability (how often to repair). |

**Figure 13. Vehicle Management System Maintainability Issues**

- Maintainability Goal: Two-Level Maintenance Operational Scenario
- Means to Achieve the Maintainability Goal: High Fault Isolation Coverage with High Level of Confidence
- Solution: Integrated Diagnostics System (IDS)

| Attributes | |
|---|---|
| **Level 1 Diagnostics:** | Derived from in-flight BIT and redundancy management |
| Capabilities: | 100% fault detection<br>100% fault isolation to functional path<br>100% fault isolation to 1 LRU for some failure modes<br>X1% fault isolation to 2 LRUs for other failure modes<br>Y1% fault isolation to 3 LRUs for other failure modes |
| **Level 2 Diagnostics:** | In-flight isolation of cannot duplicate, transient, and intermittent faults; further in-flight isolation of ambiguity faults among 2 or 3 LRUs by inferencing fault code using onboard expert system |
| Benefits: | X2% > X1%, and Y2% > Y1% fault isolation; quicker turnaround time for remove and replace |
| **Level 3 Diagnostics:** | Post-flight fault isolation using IBIT/MBIT/DBIT guided by Personnel Maintenance Aid (PMA) |
| Benefits: | X3% > X2% and Y3% > Y2% fault isolation with higher level of confidence |
| **Level 4 Diagnostics:** | Post-flight isolation by inferencing fault code using ground-based expert system and Data Transfer Cartridge (DTC)<br>Interface with the operational maintenance and logistic support data bases; build historical data base and retrieve data for fault isolation with higher confidence; perform fault prognostics. |
| Benefits: | 100% fault isolation with higher level of confidence, creating historical data base for operational maintenance, and logistic support data base use for fault isolation and fault prognostics |
| **First-Generation Integrated Diagnostic System (IDS-I):** | An environment that is constituted by four levels of diagnostics in a coordinated fashion |

**Figure 14. Vehicle Management System Maintainability Solution**




LEGEND:

RM: REDUNDANCY MANAGEMENT
BIT: BUILT-IN TEST
OES: ONBOARD EXPERT SYSTEM
PMA: PORTABLE MAINTENANCE AID
CMD: CENTRAL MAINTENANCE DIAGNOSTICS

**Figure 15. IDS Tiers for an Operational Scenario**

# METHODS TO PRESERVE THE INTEGRITY OF A COMBAT AIRCRAFT FLIGHT CONTROL SYSTEM THROUGH MAJOR UPGRADE PROGRAMMES

by
M. Rößler and W. Schmidt
MESSERSCHMITT-BÖLKOW-BLOHM GMBH
Aircraft Division
8000 München 80, Postfach 801160
FRG

## SUMMARY

During the years of in-service operation, the AFDS/TF subsystem of the PANAVIA TORNADO has gained a high level of confidence. Methods have been developed, to keep control of the integrity of the flight critical system through a plenty of modifications.

As part of major upgrade programmes of the weapon system TORNADO, several improvements will be introduced to the automatic flight control system.

It is the aim of this paper to show,

- how the new elements can be integrated into the existing system architecture without jeopardizing the integrity and availability of the system,
- how the enhanced flight control system will be validated and put into operation.

## ABBREVIATIONS:

| | |
|---|---|
| ACT | Actuator |
| ADC | Air Data Computer |
| ADI | Attitude Director Indiator |
| AFDS | Autopilot and Flight Director System |
| CSAS | Command and Stability Augmentation System |
| EMC | Electromagnetic Compatibility |
| HSI | Horizontal Situation Indiator |
| HUD | Head Up Display |
| IMC | Instrumental Meteorological Conditions |
| IN | Inertial Navigator |
| MBB | Messerschmitt-Bölkow-Blohm GmbH |
| MC | Main Computer |
| RAD ALT | Radar Altimeter |
| SAHR | Secondary Attitude and Heading Reference |
| TF | Terrain Following |
| TFR | Terrain Following Radar |
| TRN | Terrain Reference Navigation |
| TTU | Triplex Transducer Unit |

## 1. INTRODUCTION

The Multi Role Combat Aircraft TORNADO has gone into service with several airforces since 1980. An outstanding feature of this aircraft is its capability for automatic low level flight over land and over sea. The low level modes as well as the other automatically guided modes are controlled by the Autopilot and Flight Director System (AFDS), which must guarantee performance, integrity and flight safety of the aircraft during automatic operation. According to the importance of the AFDS for flight safety and aircraft performance, extensive testing is required to obtain the clearance for the system.

## 2. STRUCTURE OF THE AUTOMATIC FLIGHT CONTROL SYSTEM OF TORNADO

A block schematic of the TORNADO flight control system is shown in Fig. 1. The central part of the AFDS is the autopilot, which is a duplex redundant digital computer. The autopilot receives sensor signals from different sources according to the selected mode of operation. Pitch and roll rate demands are output to the CSAS. The level of redundancy for the various signals is indicated in Fig. 1. A more detailed description of the TORNADO AFDS is given in /1/.

## 3. AFDS CLEARANCE PROCEDURE

Extensive testing is required to get a new standard of autopilot computers cleared for in-service operation. The sequence of tests, which has been established for the TORNADO AFDS, is shown in Fig. 2.

It should be noted, that the TORNADO autopilot computers are procured as equipment which contains embedded software. The airframe company is responsible for the control laws and mode and failure logic, which represent the basic requirements for the performance within the flight control system.

The clearance procedure basically consists of the following tests:

(1) Supplier Tests: Basic equipment tests and software development tests, which are performed by the supplier on equipment level.

(2) Cross Software Tests /2/: The correct implementation of the requirements is tested by means of concurrent processing of a wide range of input patterns using original equipment in comparison with a totally dissimilar model of the autopilot computers. These tests are performed on the Cross Software Test System located at MBB. A schematic of the Cross Software Test System is shown in Fig. 3.

(3) Closed Loop Performance Tests: The correctness of the control laws and mode and failure logic and the fulfillment of the performance requirements for the AFDS is tested in the loop with original equipment, the loop being closed by a 6 degrees of freedom aircraft model. The tests are performed on the avionic/flight control rigs at MBB. Furthermore, pilot's assessment of performance aspects and familiarization with the handling of a new AFDS standard is done on the rig. The rig configuration for the AFDS performance tests is shown in Fig. 4.

(4) Flight Tests: When a new standard of autopilot computers has successfully passed the ground test procedure, it is preliminarily cleared for flight tests. After an end-to-end integration test on aircraft, a series of flight tests will be performed. The number of flights, which is required, depends on the amount of functional changes contained in the definition of the new standard.

Besides the tests, which are described above, special tests like failure investigations, safety analyses or EMC tests are part of the clearance procedure whenever applicable.

Failures or queries, which are found during the different stages of testing, will lead to iteration loops in the development process. These iterations can result in dramatic increases of costs and slippage of time scales in the course of a development project.

## 4. IN-SERVICE EXPERIENCE

The automatic flight control system of TORNADO was finally released to service in 1983. During the following years of in-service operation, the system has gained a high level of confidence. This is reflected by a continually decreasing number of reported anomalies, which has reached a constant low level till today. In Fig. 5, the tendency of reported anomalies is shown together with the appropriate number of flight hours.

The confidence in the flight control system, which has been built up through several years of operation and thousands of hours of automatic terrain following flight, is a strong impediment to every modification of the system.

## 5. MODIFICATIONS OF THE FLIGHT CONTROL SYSTEM

A plenty of new requirements resulted from in-service operation of the TORNADO AFDS:

- o New tactical requirements, e.g. to counteract an increasing threat to the aircraft
- o Requirements derived from the experience of the pilots with the aircraft
- o The basic technology is progressing rapidly during the life cycle of a flying weapon system. System updates are required to maintain the reliability and the operational use of the aircraft.

Some basic problems must be considered when new functions or equipment are implemented into a complex safety critical system like the TORNADO AFDS:

- o When the architecture of the existing system is affected, extensive clearance procedures are required.
- o Adding of new equipment necessarily affects the reliability and integrity of the system.
- o Modifications to existing equipment and aircraft wiring are expensive.
- o The operational importance of the AFDS depends on the confidence of the users in the system.

**Therefore, a trade-off must be done between operational requirements, flight safety requirements and cost aspects to define the extent of a major upgrade of the flight critical system. The number of affected equipments and functions should be minimized.**

## 6. A METHOD FOR THE INTRODUCTION OF MAJOR MODIFICATIONS

Major modifications typically consist of the following elements:

- o Redesign of existing equipment including modification of interfaces and increase of program store
- o Introduction of new equipment into the existing system architecture
- o Redesign of software, introduction of new algorithms

An attempt to simultaneously introduce all elements of a major modification of a flight control system would result in a radical change of the system, thus significantly decreasing the confidence built up during the past years and jeopardizing the results gained during the clearance process.

This discontinuity can be minimized, if in the first step all hardware modifications are introduced, while the functionality of the system remains unchanged. When the well-known system behaviour has been satisfactorily reproduced and the new hardware has been cleared to the pre-modification performance, the hardware upgrade can be introduced to the operational aircraft. Finally, the new functions will be implemented and tested to the enhanced performance limits. The final upgrade of the in-service aircraft will then be done by re-loading of software into the autopilot computers.

The following paragraph shows an example for a project, which is being carried out according to this method.

## 7. TORNADO 1st UPGRADE: MODIFICATION OF THE AFDS

At present, a major upgrade of the automatic flight control system of TORNADO is being carried out. In the following, some of the outstanding features of this modification are described with respect to their impact on the existing AFDS.

### 7.1 Description of the Modifications

#### 7.1.1 Split Axis Control

The introduction of the new mode "Split Axis Control" shall enable the pilot to control the aircraft manually in the horizontal plane, while the pitch axis is still automatically controlled by the AFDS.

In Split Axis Control, the roll stick position signal is used to generate a roll rate demand via triplex output to the CSAS. However, the roll rate demand must be automatically limited depending on the momentary roll rate and bank angle, in order to provide sufficient pitch priority and to reduce the pilot's workload.

As an operational implication of this mechanization, the pilot might apply full roll stick without being able to achieve bank attitudes as if flying in CSAS mode. In case of an AFDS auto-disconnect, there would be the danger of overcontrolling the aircraft.

To overcome this problem, adequate warnings and indications have been implemented. In case of excessive roll stick inputs, a flashing triangle is displayed on the head-up display, indicating the pilot to reduce commands.

Whereas the cockpit indications have to be supplemented for the Split Axis Control mode, the sensor data required for this modification are already available as duplex redundant information. Therefore, the pre-modification structure of the TORNADO AFDS is adequate for this new function.

#### 7.1.2 Improved Turning Flight Capability

To improve the manoeuvrability in low level flight over sea, an improved turning flight capability was demanded for the Radar Height Hold mode of the TORNADO AFDS. This requirement is fulfilled by an opening of the bank angle limiter and the roll rate demand limiter to higher values.

To reach the enhanced performance without endangering the integrity level of the automatic flight control system, it is necessary to control the aircraft angle-of-attack in the AFDS. Therefore, direct links from the already existing alpha-probes to the autopilot computers have been added to the AFDS interface, enabling a duplex redundant monitoring of the angle-of-attack. If the difference between the port and starboard signals exceeds a defined value, a monitor trip will indicate an angle-of-attack failure. The AFDS then automatically resets to a safe bank angle limitation. A new indicator is provided to display the failure to the pilot.

#### 7.1.3 Reduction of Exposure Times

The reduction of exposure times due to open loop pull-ups after system failures in the Radar Height Hold mode was required as an operational improvement of low level flying over sea.

To fulfill this requirement, the following conditions must be considered:

- o The open loop pull-up can only be suppressed for failure cases, where the aircraft can still be controlled in a safe condition.
- o Wings levelling (in case the failure occurs during a turn) must be performed under all failure conditions.

The reduction of exposure times is implemented into the AFDS by introducing a pull-up inhibit discrete signal, which prevents the system from performing an open loop pull-up after a failure has been detected, unless a pull-up voter decides, that the aircraft is in a critical situation. This voter evaluates the criticality of aircraft parameters prior to failure detection.

### 7.1.4 Improvements of Flight Director Steering Mode

In the Flight Director steering mode, the aircraft is manually controlled by the pilot according to steering information, which is generated by the AFDS and displayed on the head-up display and on the attitude director indicator.

In the pre-modification state of the TORNADO AFDS, the flight director was a simplex facility and could therefore not be cleared for low level IMC operation.

Flight director steering operation in low level IMC requires cross- monitored duplex calculation of the flight director algorithms. This is accomplished with the TORNADO 1st Upgrade by cross-monitoring and signal generation of normal acceleration demand and bank angle demand in the same way as for fully automatic flight control. If one of the AFDS computers fails, simplex operation of the flight director is available as reversionary mode. This case has to be signalled to the pilot by adequate warnings.

In case of a processor failure in one of the two AFDS computers, both automatic and manual cross-monitored flight control cannot be used, as they are not independent from each other. On the other hand, the use of existing hardware for the implementation of a cross-monitored flight director into the TORNADO AFDS is an economic way to enable practising of manual flying in IMC conditions.

## 7.2 Stepwise Introduction of the AFDS Upgrade

### 7.2.1 Introduction of New Hardware

The 1st Upgrade modification to the TORNADO AFDS consists of the following elements:

- o Redesign of the hardware of the autopilot computers. Growth potential of program storage and provision of space and wiring for a MIL-Std 1553 B interface has been included in the hardware redesign.
- o Extension of the interface of the autopilot computers to connect the alpha-probes.
- o Extensive change of the control laws and of the mode and failure logic of the autopilot computers.

The upgrade is being introduced in two steps (see Fig. 6). In the first step, the pre-modification performance of the AFDS has been implemented on the upgraded hardware.

To get the new hardware cleared for production with the pre-modification functions, a limited clearance procedure was accomplished including 25 hours of closed loop simulation on the rig and 10 test flights. A read-across of the test results showed, that the aircraft performance had not changed compared to the pre-modification state of the AFDS. The new hardware standard is now released for installation into series aircraft.

### 7.2.2 Clearance of the New Functions

Presently, the new AFDS software is being developed. As major changes of the aircraft performance in the safety critical low level modes are introduced, extensive tests are required to clear the modified system.

After two iterations of the ground test procedure, the first software release was cleared for experimental flights. A total of 150 hours of performance testing on the rig resulted in 20 software queries. During cross software testing, two dormant failures have been found in the autopilot software. With the first flight-cleared AFDS standard, 30 test flights have been performed, resulting in 4 queries and several modifications of operational requirements. None of the queries reported during flight test has been rated critical. Analysis shows, that most of the problems found during flight, could not have been detected by rig simulation.

At present, the next issue of AFDS software is being prepared, which will incorporate all queries and requirement changes reported so far. A total of 100 test flights will be required to get the upgraded AFDS functions finally cleared. When the tests have been successfully completed, the new software, which is stored on EPROM in the autopilot computers, will be reloaded to the in-service aircrafts already operating with upgraded AFDS hardware.

## 8. IMPLEMENTATION OF TERRAIN REFERENCE NAVIGATION IN A FURTHER TORNADO UPGRADE

At present, a Terrain Reference Navigation (TRN) mode is being developed for the TORNADO flight control system. In this mode, the flight path of the aircraft is controlled by comparing the measured terrain height profile with a digitized map stored in the TRN computer. The height profile is calculated as difference between the vertical channel of the inertial measurement unit and the height above ground measured by the radar altimeter. The aircraft can operate with a significant reduction of radar emission compared with the present Terrain Following mode. A detailed description of the TRN system is given in /3/.

### 8.1 Hardware Aspects

There are two different ways of interfacing the TRN computer to the existing TORNADO flight control system, which must be investigated with respect to system integrity aspects:

- o Direct interfacing of the TRN to the autopilot computers (Fig. 7a)
- o Interfacing TRN and forward-looking radar to the autopilot computers via a switching unit (Fig. 7b)

The direct interfacing method uses the redundant AFDS design, which is in agreement with the fault tolerance requirements for safety critical flight control functions, for the implementation of the monitoring and mode switching logic. The integrity of the existing Terrain Following function is fully preserved. On the other hand, a modification of software and hardware of the autopilot computers would be necessary.

When using a switching unit as interface between TRN and AFDS, the autopilot computers could remain unchanged. However, a new equipment would be introduced into existing signal paths, influencing reliability and integrity of the flight control functions.

### 8.2 Stepwise Introduction

Flight guidance using TRN requires a precise determination of the aircraft position and relies on map data, which have been loaded on ground. A major hardware modification is needed for the integration of TRN into the existing system architecture.

The introduction of a new terrain following mode, which does not make use of an active forward-looking sensor, will result in a significantly decreased level of confidence. To build up confidence into the new elements, the TRN function must be introduced step by step.

In the first step it must be shown, that the integrity and the performance of the established terrain following mode (guided by the forward-looking radar) is not decreased by the integration of the new hardware elements.

In the second step, the g-commands derived from the TRN data base will be monitored by the guidance information from the forward-looking radar, thus enabling fail-safe operation with respect to the new mode. Only if the monitored mode has been cleared, the silent TRN mode can be tested, gradually decreasing the clearance height.

## 9. CONCLUSIONS

During the in-service life of a flying weapon system, which is of the order of 30 years, major system upgrades cannot be avoided.

Based on presently running and intended upgrades of the TORNADO flight control system, it has been described, what measures must be taken to preserve integrity, fault tolerance and performance of the existing system during a major upgrade.

A method has been outlined, how to introduce a major modification without jeopardizing the confidence into the system, which has been built up during years of in-service operation.

## REFERENCES:

/1/ W. Schmidt and U. Butter:
"TORNADO AUTOPILOT, MEASURES TO ENSURE SURVIVABILITY AFTER FAILURES"
AGARD Conference Proceedings No. 347

/2/ J. Stocker, J. Rauch:
"CSTS: EIN SOFTWARE-TESTSYSTEM FÜR DEN TORNADO-AUTOPILOTEN"
MBB/FE325/S/PUB/118

/3/ M. Eibert, P. Lux, G. Richmond, A.J. Henley:
"SPARTAN/ISS - A Combined Terrain Topography Referenced Navigation and Terrain Following System"
presented on the 48th Symposium of the AGARD GCP

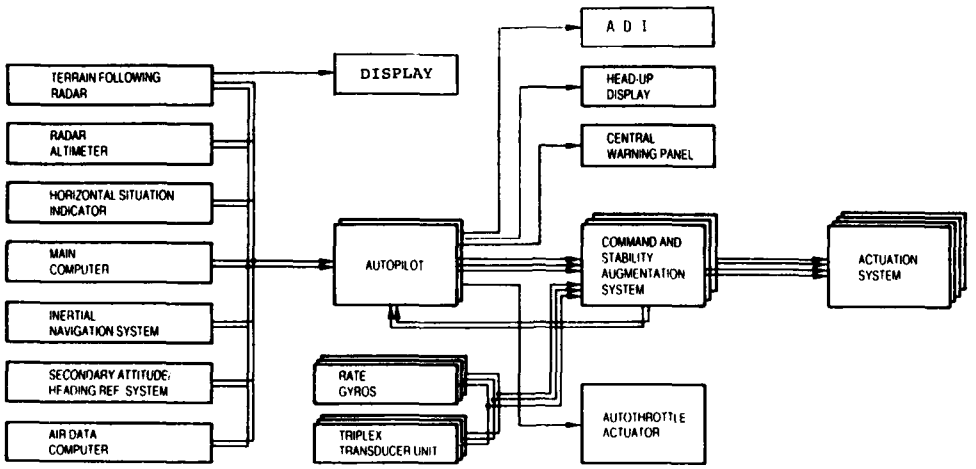



Figure 1: Block Schematic of TORNADO Flight Control System

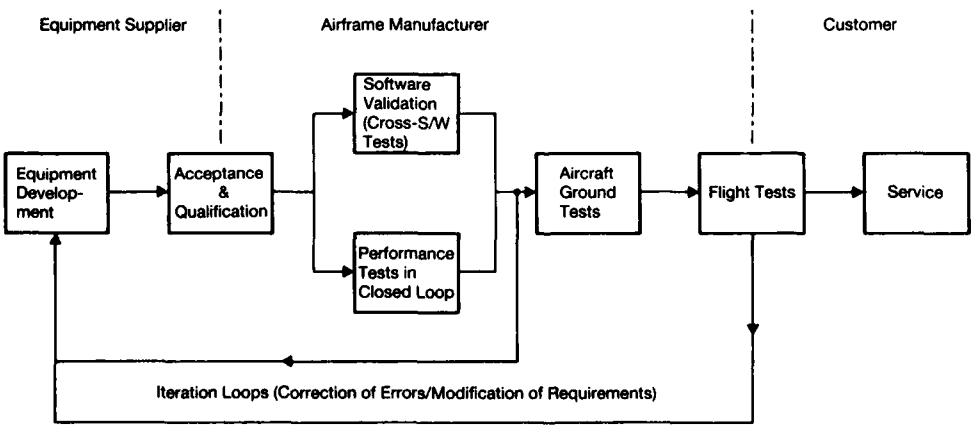



Figure 2: Sequence of Clearance Activities for Modifications of the TORNADO AFDS

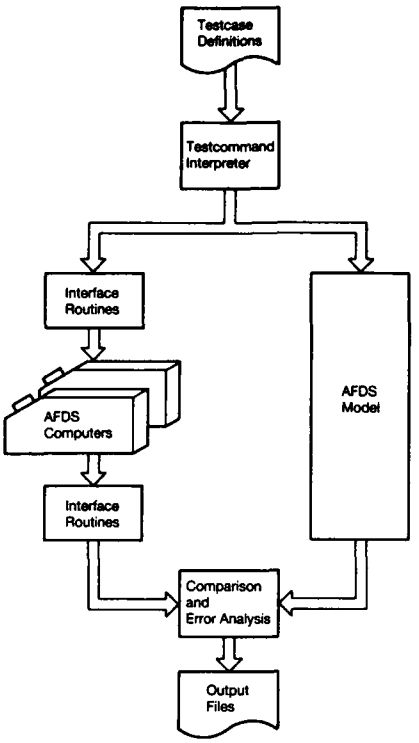



Figure 3: Block Schematic of the Cross Software Test System

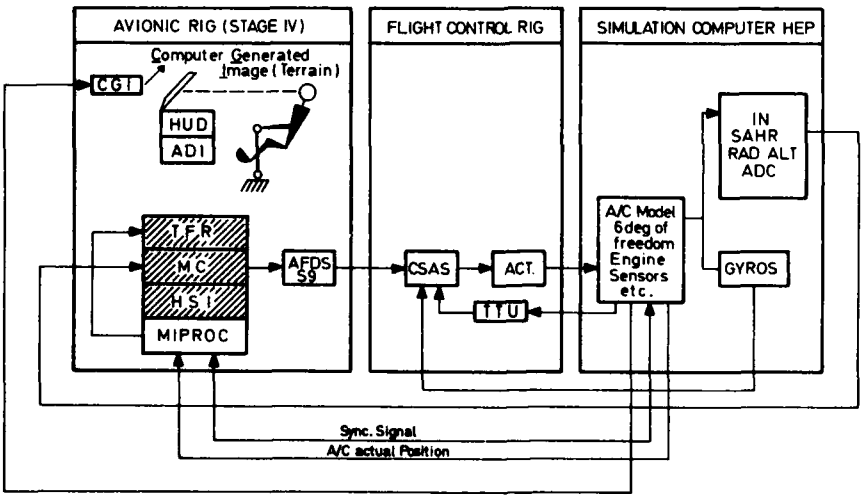



Figure 4: Rig Configuration for AFDS Performance Tests

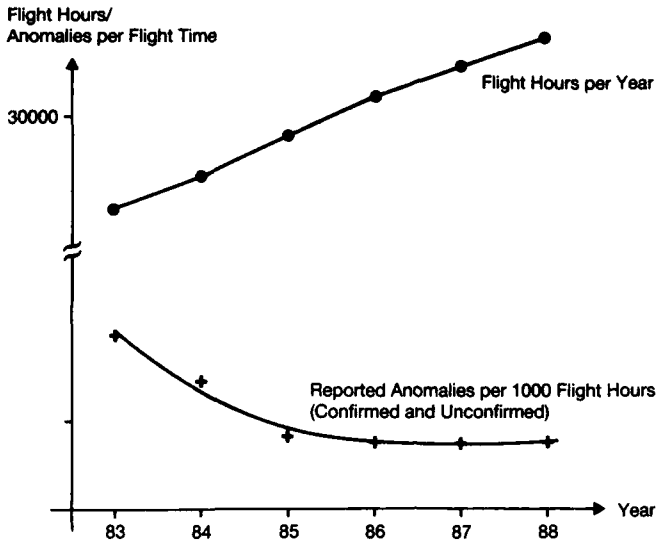



Figure 5: TORNADO AFDS In-Service Experience.
Tendency of Reported Anomalies together with the appropriate number of flight hours (for selected services).

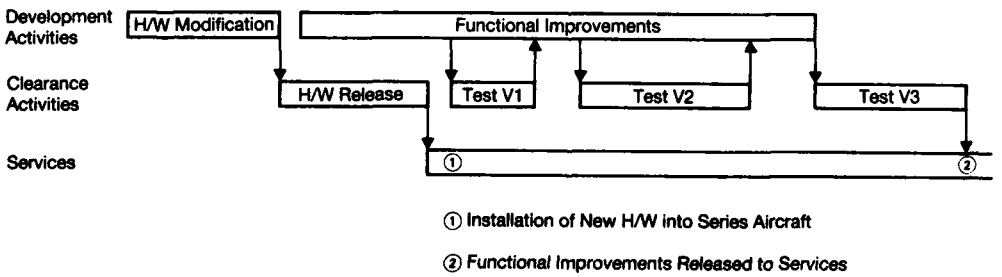



Figure 6: Stepwise Introduction of the TORNADO AFDS Upgrade

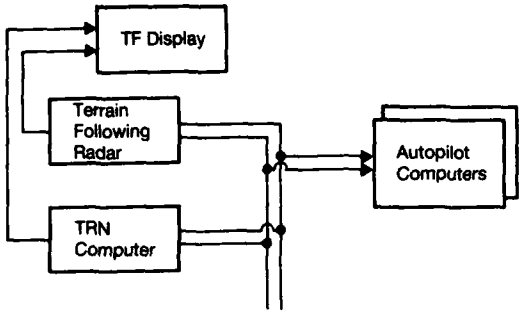



Figure 7a: Direct Interface TRN/AFDS

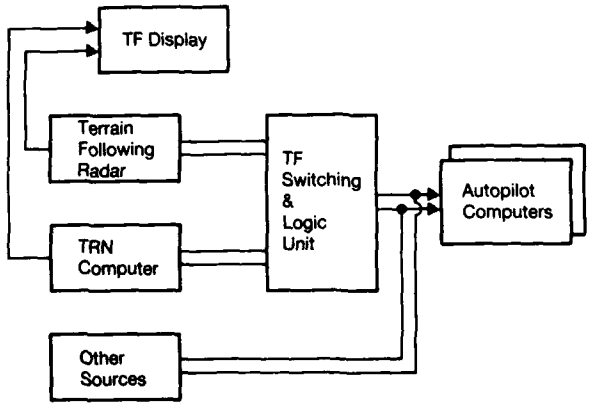



Figure 7b: Interface TRN/AFDS via Switching/Logic Unit

# RESEARCH INTO A MISSION MANAGEMENT AID

by

J.R.Catford (GEC Avionics) and I.D.Gray (Ferranti)
both of the MMA Joint Venture, RAE Farnborough
Hants GU14 6TD
United Kingdom

## 1. Introduction

Pilot workload in fighter aircraft is ever increasing, driven by the need to fly low and the complexity of the systems and weapons in modern aircraft. The density and growing sophistication of Warsaw Pact anti-aircraft weapon systems greatly exacerbate the problem.

The current European trend towards single seat fighter aircraft, on grounds of system and manpower costs, increases the workload problem still further.

The research project, reviewed in this paper is directed to put the crew back in charge by introducing new levels of weapon systems automation.

The Mission Management Aid (MMA) is scheduled for clearance into service in the first decade of the next century and although it is confidently expected that the airborne computing power to perform the task will be available in that timescale, the complementary disciplines to design, test and validate such a system will need to be developed. The current research programme is concentrated on the timely development of these disciplines.

The basic functional approach to the MMA was outlined some four years ago, by a group of senior engineers drawn from British Aerospace, GEC Avionics, Smiths Industries and Ferranti Defense Systems. Over the same period scientists at the Royal Aerospace Establishment, Farnborough were examining similar approaches.

Research into the MMA is being undertaken jointly by the four industrial organisations and the Royal Aerospace Establishment. The current programme which has been running for two years, involves seconded staff from the industrial organisations and the RAE, Farnborough where the team is located.

It is very evident that current fighter aircraft systems are composed of a number of well tested evaluated and validated sub-systems and yet the integrated system tends to exhibit serious operational problems and a long time delay before modifications to clear these problems can be introduced. It is hence essential that the more complex systems of the next century exhibit characteristics that allow for rapid modification to meet changing operational needs and yet maintain essential operational integrity through these changes.

## 2. The Joint Venture Organisation

The Joint Venture set up to prosecute the development of the MMA is regarded as unique, at least in terms of the United Kingdom Defence Industry. The four industrial organisations have set up a Joint Venture Agreement with the Royal Aerospace Establishment of the British Ministry of Defence (MOD) on the basis of equal sharing of both costs and benefits by the five partners.

The basic Organisational Tree of the MMA Joint Venture is shown at figure 1.

The Joint Venture Team is housed in modern accommodation at RAE Farnborough and staffed by twenty systems engineers drawn equally from the five organisations.

The Team is well supported by modern Work Station Equipment, illustrated in figure 2.

The primary objective of the MMA Joint Venture is the development of a real-time multi mission, multi scenario simulation of the MMA.

The major milestones in the programme to this end are illustrated in figure 3.

The preliminary study phase was completed in April 1988 and the Programme Phase commenced in June of that year. The current Prototype Phase is set to functionally explore all the major aspects of the MMA in non real-time on the Work Station Network.

## 3. Prototypes

The prototype work is currently occupying ninety per cent of the Team's efforts, and will provide an evolving Prototype MMA hosted on the twelve networked work stations.

The majority of the Prototype activity is written in Common Lisp and hosted on the Symbolics Work Stations. Specialist activities such as graphics intensive displays are run on the Silicon Graphics and the Sun.

Prototype simulations of the MMA on the work stations will run at about twenty times real-time.

Functional Specifications derived from prototype software will lead to refinement of the functionality of the prototype and form the specification basis for the functionality of the Mission Capable Simulation of the MMA.

## 4. The Need For An MMA

The requirement for a MMA has already been referred to in terms of pilot workload. The drivers will now be identified in more detail in terms of one of the missions that has been identified as suitable as an example mission for the prototype work;

* Air Interdiction 100km beyond the Forward Edge of Battle Area (FEBA).

### 4.1 Air Interdiction

The phase of an Air Interdiction Mission from a high level systems engineering viewpoint can be classified as:

* Mission Planning
* Mission Briefing
* Briefing Update
* Take Off
* Friendly Ingress
* Hostile Ingress
* IP to target
* Escape
* Hostile Egress

etc.

We will concentrate on the Hostile Ingress Phase as this phase clearly indicates many of the drivers that lead to the need for an enhanced class of Mission System automation, exemplified by the MMA.

The primary objective of the Hostile Ingress phase is to arrive at the Target Initial Point safely with maximum fuel, maximum disposables (chaff, flares etc.), undetected and on time.

The achievement of this objective is hindered by dense ground to air defences and air to air threats. The first defensive measure is to fly low and fast but this is hindered by the manoeuvrability and drag limitations imposed by a typical external weapon load.

The primary consideration during the phase is self defence. At a ground speed equivalent to about 1km every three seconds unexpected threats may be arriving fast. Whilst combating these threats the pilot has to fly, navigate and communicate. A high workload situation applies throughout the hostile ingress and the pilot is liable to go into workload saturation at critical moments. The MMA will reduce the workload to an acceptable level by enabling automation of lower level functions and providing tactical options which on selection by the pilot will be automatically executed.

## 5. The Air to Ground and Air to Air Scenarios

### 5.1 General

A range of missions and threat environments were considered as potential scenarios for the employment of MMA equipped aircraft over the next several decades. These covered both the air to ground and air to air environments as it was considered that in the future both the pilot and the aircraft will be considered multi-role, and that specific missions could well encompass aspects of both roles. The scenarios were considered from a workload point of view both for the MMA and the pilot. The other driving consideration for prototyping the MMA being how "controlled" (or pre-planned) the operating environment was both for testing and comprehension purposes. The scenarios reviewed are summarised in figure 4.

### 5.2 Air to Ground

There were two broad categories of mission considered:

a. Close air support - this involves co-operation with ground forces to provide air cover and short range ground attack capability. It is characterised by the employment of "smart" weapons possibly under ground control (especially where there are troops in contact and there is a forward air controller suitably positioned), and the short time scales and consequent lack of pre-planning of targets.

b. Interdiction - this is a short to medium range ground attack mission against a pre-planned target carried out by a small group of aircraft. It is a mission demanding minimal use of active sensors, in order to remain unobserved as long as practicable, co-operation between aircraft, and a high degree of pre-planning of all mission phases to and from the target. Consideration of a long range (500 kilometres plus) interdiction was limited as it was felt that in the timescale of the Joint Venture the format of such missions would undergo significant change.

Although both missions had a high level of pilot workload, a 100 kilometre interdiction at 100 feet and 500 knots was chosen as the initial prototype scenarios as it would most fully exercise the MMA's functionality.

5.3 Air to Air

Here again, although the missions are more difficult to delineate, there were two major categories:

a. Interception - here there are (usually) pairs of aircraft executing a combat air patrol (CAP) at 5,000 to 30,000 feet alternately scanning for airborne threats or targets. On detecting such they are pulled off CAP to carry out a point interception on the raid and its possible escorts. It is a co-operative exercise between the CAP aircraft demanding good communications, the use of joint tactics and the maintenance of adequate situational awareness. Pre-planning can only be done to the extent of general intelligence briefings as this is very much a dynamic re-planning mission.

b. Suppression - this is much more difficult to define as a single mission but is involved in the establishment and subsequent maintenance of air superiority. It encompasses fighter sweeps and the suppression of enemy air defences, and, although basically of short time duration, it can involve a significant amount of pre-planning.

The suppression missions have a higher sustained workload both for the pilot and MMA, but both types are carried out in a generally less controlled environment than the air to ground missions and are being studied for their potential implications on MMA development.

6. Architecture of the MMA

6.1 General Overview

The MMA is basically a tactical advisory system for the aircrew of military aircraft and as such consists of a planning core, the pilot support and interaction facilities, and all the subsidiary aircraft system support and interface functions. The high level architecture of the MMA is related to the other aircraft systems and to the pilot and his environment in figures 5 and 6.

6.2 Core Functions

The core of the MMA consists of two major functional blocks:-

a. Pre-processing - the data fusion and situation assessment functions for pre-processing aircraft sensor data, intelligence and pre-mission brief data into a form that can be used by the planner.

b. Planner - the functions that produce tactical plans based on mission objectives, the current situation and currently available resources such as weapons and countermeasures.

6.3 Man-Machine Interface (MMI)

The interaction between the pilot and the MMA is crucial as the MMA is a tactical aid, and the pilot is always in control of the aircraft. Hence the MMI has involved a great number of human factors studies resulting in a design for two functional display surfaces:-

a. Immediate - everything that will affect the pilot in the next 10 seconds of flight displayed in head-up format.

b. Plan - everything that will affect the pilot in the next several minutes of flight displayed in a head-down format.

To provide these interface formats requires two major functional units within the MMA.

a. Pilot interface manager/controller - contains all the pilot/MMA interface functions with a three tier bus structure (immediate, plan and message) for data handling. Internally it consists of a prioritisation function and a scheduler to decide on the relative importance of the various pieces of information needing the pilots's attention before they are displayed.

b. Display controller - a fairly straightforward format generator for the plan and immediate displays.

### 6.4 Emergency Response

Consideration has been given to emergency response functions that automate reaction to situations in which pilot reaction time is inadequate.

An example is the imminent approach of a missile where all defensive measures appear to have failed and the aircraft is at immediate risk. Such functions have obvious integrity implications and in addition require a detailed understanding of the emergency functions. Much of this understanding will be obtained as the basic advisory functions of the MMA are explored and as a result detailed exploration of the emergency functions will occur later in the project.

Nonetheless the integrity implications of the emergency functions will be kept under continuous review as the programme progresses.

### 6.5 Managers

These three functions are basically resource schedulers and housekeepers for the aircraft systems most intimately connected with the core MMA.

a. Sensor - processes requests from the core MMA for extra information from the sensors. It checks the relevance of a sensor for the request, the availability and the possibility of reconfiguring the sensors to cope with all requests on a prioritised basis. Finally commands are issued to the affected sensors.

b. Navigation - performs terrain referenced navigation feeding into terrain following/avoidance which also processes threat avoidance information from the core MMA. In addition it controls the navigation specific sensors such as the inertial platform.

c. Communications - controls the communications systems as well as providing Emission Control (EMCON) data to control the active/passive use of sensors important for stealth operations.

### 6.6 Status Monitor

All the aircraft systems provide levels of status information as well as general health reports. The MMA must continually monitor these reports and provide this information to the pilot in a meaningful way. In the case of health reports this will involve a prioritisation function to decide on the urgency and criticality of the report based on the mission phase and in the light of other reports. This information along with general status data is then made available to all the relevant MMA functions and for presentation to the pilot after assessing exactly what is important for him to know at this mission phase. Finally the data is logged for post-mission analysis purposes.

### 6.7 Peripheral Support

This provides for the ultimate implementation of the tactical plan for the next phase of the mission on its approval by the pilot. It basically provides the interfaces through a tactical implementation function to the aircraft systems such as the flight control and stores management systems which are not directly connected through the above manager functions.

## 7. Organisation of the Core Functions

### 7.1 General Functionality

The general organisation of the core functions and their interfaces is shown in figure 7.

The core of the MMA's operation is concerned with producing an "optimal" tactical plan for pilot selection from a group of options. This is a three stage process concerned with taking data from a number of sources and combining this to produce a meaningful single view of the outside world. Combining this view with "intelligence" and pre-mission brief information and placing value judgements on this data enables production of an assessed view of the situation in the light of the current and future phases of the mission. Finally combining this assessed view with mission objective information results in a number of tactical options - the plans.

### 7.2 Sensor Fusion

This takes in information from the aircraft sensor systems, communications and the terrain database, and processes it in two stages into an alpha scene (a view of what the aircraft can see in the outside world with associated confidence intervals). The first stage is the correlation of tracks into positions and, where possible, velocities. This involves the alignment of data from sensors with different accuracies, temporal and spatial references, frames, and the subsequent association of tracks into a single resolved track with a confidence interval. The second stage is attribute fusion, the identification of targets using sensor data in the form of RADAR or Infra Red (IR) signatures, along with contextual and historical information, to separate and identify targets which are possibly spatially indistinguishable. Statistical filtering techniques are applied in the first

stage while evidential reasoning is used in the second. The output from fusion is the alpha scene consisting of a list of outside world "objects" with their positions, velocities and identifications, where each quantity has an associated confidence interval.

7.3 Situation Assessment

This is a filtering process applied to the alpha scene to produce a (potentially) much smaller beta scene, which only contains a prioritised list of objects rather than the entire outside world view. It is a multistage process, continuously reiterated, in which objects known to be friendly at this stage are first filtered from the scene for separate processing because, although they do not constitute a threat, their presence can influence the overall assessment of the threat environment. The remaining objects, hostiles and unknowns, are evaluated for threat and target potential, where the values are of two types:-

a. Static - an inherent value dependent on the identification of the type of threat/target and its current relation to the MMA aircraft.

b. Dynamic - a weighting factor applied to the inherent value to indicate how the threat/target interacts with the current tactical plan (gamma option), for example relative target aspect. This demands a high degree of feedback between planning and situation assessment, and this is catered for by splitting situation assessment into two processes, the second of which is closely related to the planning function and the current plan.

Having been evaluated the objects are then prioritised with respect to the current tactical plan, and finally those adjudged to be the most important, (threatening, vulnerable or supportive), are filtered off to make the beta scene. The output beta scene has the same format as the much larger alpha scene with positions, velocities and identifications for each object within the scene, but with the addition of a "threat" value and order of priority, and the masking of the uncertainty attached to the original reports.

7.4 Planning

This is the heart of the MMA which constructs tactical plans (gammas) including a gamma* option, that perceived to be the most favourable. Its plans are built from the beta scene (current situation) input, current constraints and the mission objectives obtained from the pre-mission brief. The final gamma* output has several parts covering, for instance, the employment of weapon and countermeasure systems, and the tactical route generated by the threat avoidance function, which are fed on to the appropriate aircraft systems. Three of the major functional blocks within the planner are:-

a. Objective response manager - constructs the tactical plans (gammas) in the form of a multilevelled tree whose entries represent assessed values for that stage of the mission following that particular plan. A search is then performed through this tree structure for the best option, the gamma*, using search techniques appropriate for dealing with adversaries. The gamma* is then output to the pilot to accept or reject before being passed on to the relevant aircraft systems.

b. Attack and countermeasure options - based on the mission objectives, values of potential targets and threats and the current status of the aircraft's weapons and countermeasures, this evaluates options for an attack/defence strategy to be incorporated in the gamma tree constructed by the objective response manager.

c. Tactical routeing - is an airborne small scale rerouteing function for threat and terrain avoidance applied at a deep level within the gamma tree. It constructs a threat cost matrix incorporating a coarse level of terrain avoidance and then performs a constraint governed search on this. The output is then in the form of a list of threat avoiding waypoints which can be further processed by the navigation systems manager.

7.5 Interfacing

The core function interfacing is primarily concerned with the data structures used to link the various internal and external functions together and with how and what structures are presented to the pilot.

a. Functional - the primary internal interfaces are the alpha scene to link the fusion and assessment functions, and the beta scene to link the assessment and planning functions. Both are held in the form of list/trees containing positional, track and identity information with, where appropriate, confidence and object priority values. Reports from the aircraft sensors are fed in a variety of list structures, and the plans (gammas and gamma*) are fed out as trees where each successive level is associated with a greater degree of refinement for any given mission phase.

b. Pilot - most of the time the pilot will interact with the assessed view (beta scene) and the plan (gamma*) presented on the plan and immediate displays above. However, dependent on his workload, he will always have the option to interrogate deeper into the alternative plans (gammas) and the outside world view (alpha scene). This ability is important both to allow the pilot to fully appreciate the pros and cons of the alternative plans and for building up a level of confidence in the MMA's operation.

## 8. Integrity Considerations

The MMA will inevitably be implemented in a distributed hardware architecture and complex interrelated software will be targeted to that hardware. Much of the software will be difficult to test and validate using conventional techniques because optimal solutions to the problems the MMA is solving are not generally available.

At the lower modular levels the software will be amenable to conventional test and validation techniques, at the higher levels new approaches will be required. Consideration of these matters is still at an early stage on the MMA Joint Venture and the problems are endemic to any mission software of the level of complexity of the MMA.

Certain stages have to be isolated in the route towards validation of software of this class.

a. Flight critical elements must be isolated and tackled using techniques appropriate for flight critical systems.

b. If the mission functions are hosted on a modern implementation architecture with hardware incorporating such features as initiated self test, continuous self test, maintenance data access, and a degree of spare capacity to overcome first failures by limited reconfiguration, then the validation of the mission software is the critical item.

Rapid Prototyping of the MMA functions will lead to a hierarchy of Functional Specifications. These Specifications will be used to develop a real-time Mission Capable Simulation of the MMA. Extensive testing of the Mission Capable Simulation will result in changes to the Functional Specifications and retesting.

Full Scale Development of an MMA for a specific application can therefore proceed on the basis of a good definition of the functional requirement. Implementation software on the target hardware can be validated against these functional requirements.

The ruggedness of the system can be assessed by deliberate "edge of envelope" stress testing.

The final stage of assessing the operational utility of the system will require extensive flight testing backed by comprehensive simulation and rig facilities.

Finally it must be noted that the problems outlined above are not new. Many of the current in service aircraft have digital mission systems composed of extensively tested and validated hardware modules and software suites. Nonetheless the working up of these aircraft to full operational capability often involves numerous software changes before the mission systems meets operational needs. Many of these changes cost significant time and money.

Though a number of techniques borrowed from the artificial intelligence community are being investigated and prototyped in the MMA programme, it is important to emphasise that validation and certification of the MMA is a specific concern. Hence the design aim for the MMA will be such that for precisely equal input data sets over time the MMA will produce identical results.

Satisfactory achievement of this design aim will aid the psychology of validation and certification but will neither make the MMA tactically predictable or reduce the trials and certification effort required. The quantity and variability of the data inputs in practical mission conditions will ensure "unpredictability".

Though the Joint Venture approach to the MMA should result in a good array of Functional Specifications as a starting point to full scale development of the system the need to allow flexibility in the organisation of the system to allow rapid change to meet operational needs will be of extreme importance.

## 9. Outline of the Future Programme

Whilst the MMA Joint Venture is proceeding apace on the Work Stations selected for the Prototype, essential elements of validating the basic concept depends on real-time operation. This is particularly true for the Man Machine Interface aspects of the work. The MMA is designed to aid the pilot and maintain workload at an acceptable level in the most critical situations.

The first trials of the MMA in a real-time environment are scheduled for 1992 and will be directed to exploration of the crew interaction with the MMA. These trials will be hosted on a Mission Simulator in RAE's Mission Management Department and will concentrate on simulated low level flights over a model board so as to provide an intensive work load for the simulation "pilot".

## 10. Conclusions

The MMA Joint Venture is in its early stages, but the Prototyping phase is now providing clear insights into the future development of the MMA. The later Mission Capable

simulation phase will initially concentrate on pilot interface aspects and provide the opportunity to examine the critical interface between the pilot and this new class of system automation. Though the MMA will create new opportunities for the organisation of the Man Machine Interface in fighter aircraft it is essential that an evolutionary approach is taken in order that the operational MMA properly complements the pilot.

The validation and certification of a complex system like the MMA will require the development of new techniques which must, again, be approached in an evolutionary way.

ILLUSTRATIONS




PARTNER ORGANISATIONS

British Aerospace(MAD)
GEC Avionics & Sensors
Ferranti Defense Systems
Smiths Industries Aerospace & Defense Systems
Royal Aerospace Establishment, Farnborough

Figure 1 Management Organisation

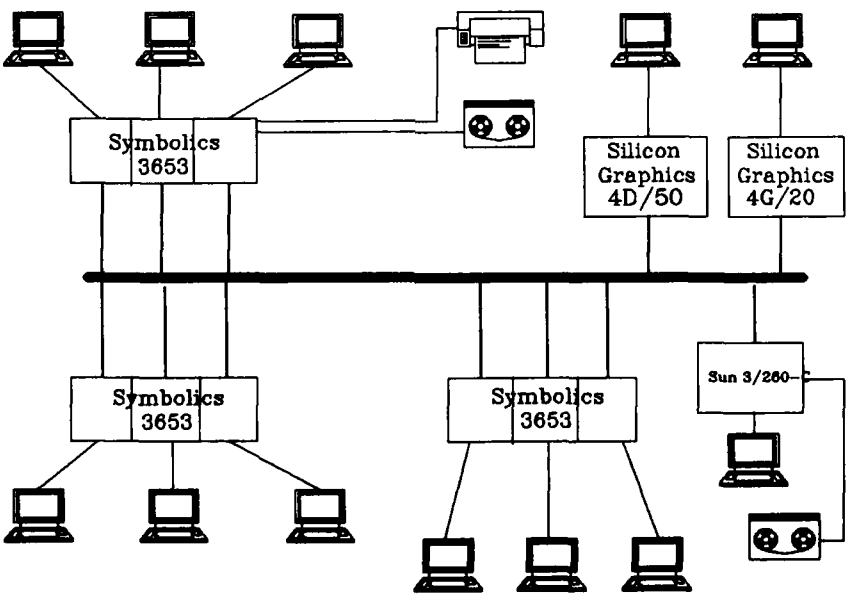



Figure 2 Work Station Network

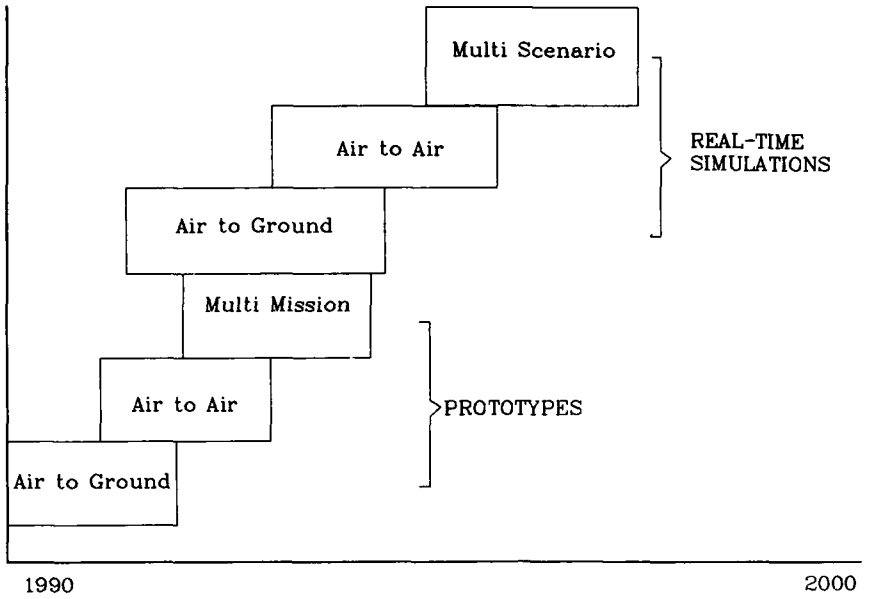



Figure 3 Programme Milestones

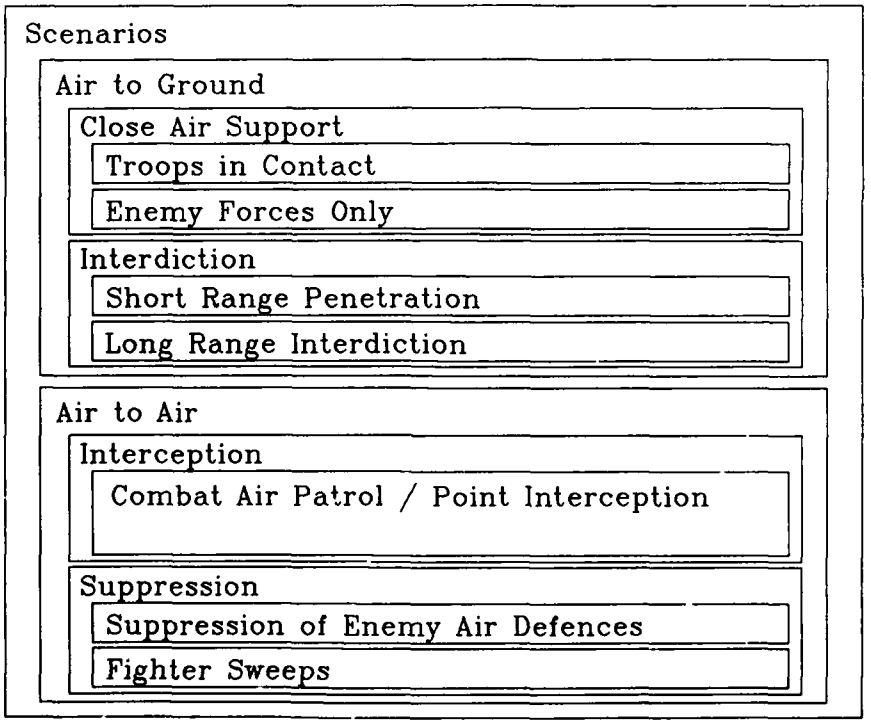



Figure 4 Scenarios Considered

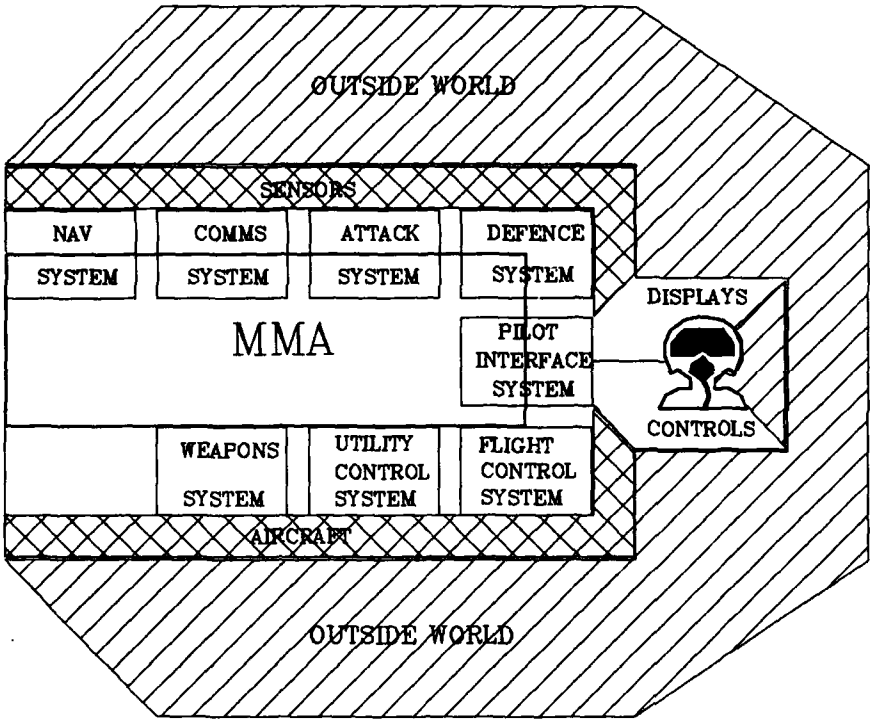



Figure 5 The MMA as an Aircraft System

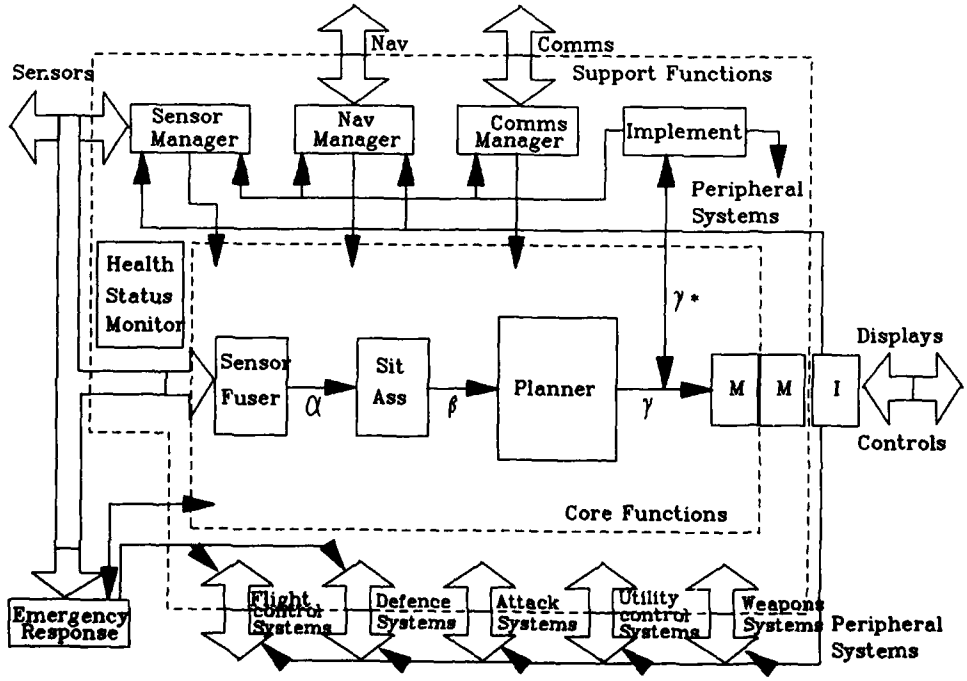



Figure 6 MMA Architecture

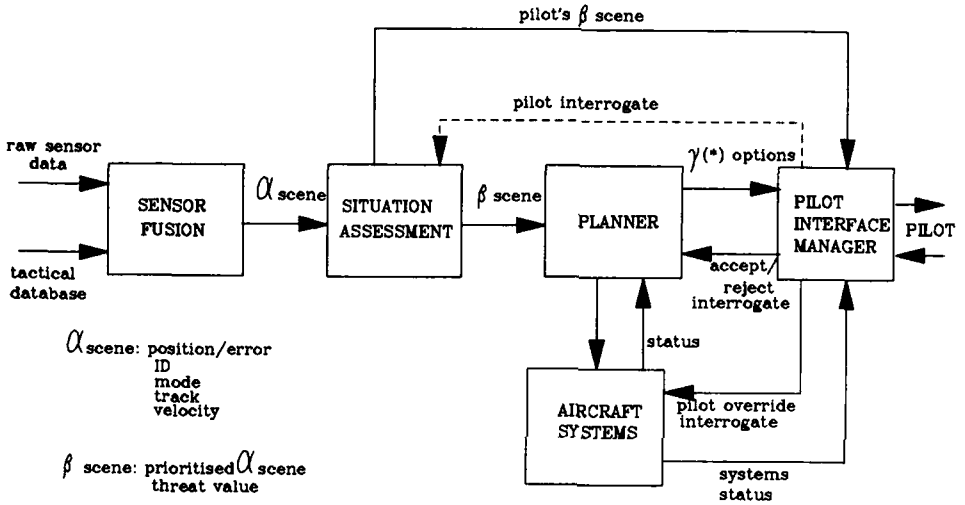



**Figure 7 MMA Core Functions and Interfaces**

# Integrated Diagnostics for Fault-Tolerant Systems

Harry A. Funk
Mark M. Jeppson

Honeywell Systems and Research Center
Minneapolis, Minnesota 55418, U.S.A.

## Summary

This paper offers an integrated approach to the maintainability of high-reliability fault-tolerant flight control systems. Modern aircraft provide designers of maintenance systems a tremendous amount of data on the health of subsystem elements. Examples include initiated built-in-test, continuous built-in-test, redundancy management status, reconfiguration status, and time-stress measurement data. Advances in both on-aircraft and off-aircraft diagnostic hardware and software provide the designer with a wide range of partitioning options to most effectively use these data. This paper discusses an integrated maintenance approach using both a portable maintenance aid at the flight line and on-aircraft in-flight diagnostic resources. An implementation strategy for each of these systems is presented along with a technique that ensures designed-in commonality between the on-aircraft and off-aircraft systems. The proper use of these systems in addressing particular maintenance problems (re-test okays and cannot-duplicates) is discussed.

## Integrated Diagnostics Goals

Since the early 1980s the term *integrated diagnostics* has been used to stress the importance of defining a process and the interfaces required to merge all of the information generated in the course of diagnostics. The ultimate goal is to provide a cost-effective capability for the detection and unambiguous isolation of all faults known or expected to occur in a weapon system. For the purposes of this paper, *integrated diagnostics* denotes "a structured process that attempts to maximize the effectiveness of diagnostics by integrating the management and delivery of all diagnostic support elements to provide a cost-effective capability for the detection and isolation of all faults." These support elements include built-in-test (BIT), automatic and manual test equipment, technical documentation, training, manpower, and maintenance aiding. A simple conceptual view of integrated diagnostics is illustrated in Figure 1.

For the integrated diagnostics concept to maximize the effectiveness of the combined elements, the structure imposed on the various elements not only must provide for data sharing between the various elements of the system, but must do so in a non-overlapping and complementary fashion.

Given this statement of the goals of the integrated diagnostics concept, the next step is to examine the resources available to support, and the constraints that impede, the achievement of these goals.

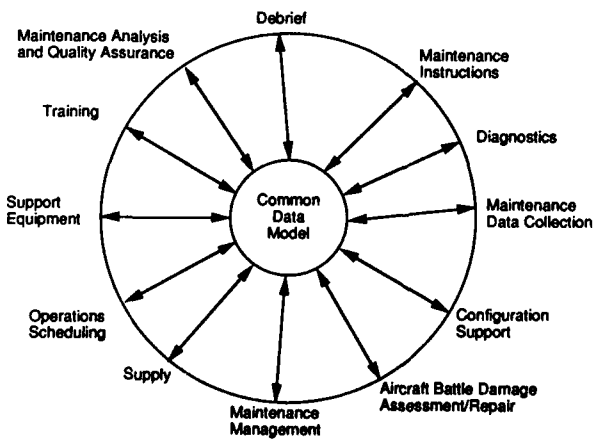



**Figure 1. Conceptual View of Integrated Diagnostics**

This research was supported in part by Wright Research Development Center Contract No. F33615-85-C-3613.

## Resources

Items that aid an integrated diagnostics process can be categorized as either on-aircraft or off-aircraft. Elements of these systems range from those designed within the system (embedded diagnostics) to special elements required for off-aircraft support (automatic test equipment).

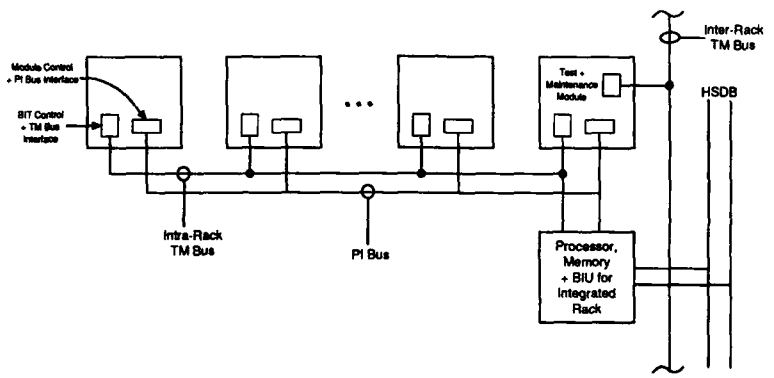
Whether these elements are on-aircraft or off-aircraft is of little concern. The key point is that they are part of the total support structure that is designed and integrated as part of the weapon system.

## On-Aircraft

Fault-tolerant architectural concepts that have surfaced as a result of USAF-funded contracts such as Multi-Function Integrated Sensor Suite (MFISS), PAVE PILLAR, and Flight Control Maintenance Diagnostic System (FCMDS) provide examples of diagnostic features designed within the system. Figure 2 provides an illustration of the BIT capability for an integrated, modular avionics approach. This architecture illustrates a maintenance/diagnostic system (MDS) which is independent of the primary hardware functions; this prohibits faults from propagating between subsystems, which is critical when considering systems such as flight control. The system is also accessible from the exterior to the line-replaceable module (LRM); this provides the opportunity for complete and timely visibility into the subsystem.

This diagnostic system has a hierarchical organization consisting of three levels:

- (Aircraft) system level
- (Integrated rack) subsystem level
- (Card) module level




- BIT may be initiated at power-on by external request continuous bit possible.
- BIT command/response passed between integrated racks by inter-rack TM bus.
- BIT command/response passed within integrated rack by intra-rack TM bus.
- BIT is controlled as a full time task of the test and maintenance module.
- Design is extensible and does not require rack controller, PI Bus or High Speed Data Bus resources.
- Failure logging and reporting do not depend on computational or communication resources of the primary equipment. A failure in the primary equipment can still be detected, isolated, logged, and reported.

**Figure 2. Built-In-Test Philosophy Based on a PAVE PILLAR Architecture**

This organization is shown in Figure 3. At each level, subordinate diagnostic system processors communicate with their superiors over a dedicated test and maintenance bus. The system diagnostic processor (SDP) supervises the diagnostic processing of the total avionics suite. The SDP diagnoses faults to the LRM level and provides mass storage for archiving diagnostic system processing results. It also provides an interface to a maintenance technician through a maintenance panel that can be either an integral part of the system or a small carry-up device the size of a lap-top computer.

The subsystem diagnostic processor (SSDP) is itself a LRM that is integrated into each rack. The SSDP controls the diagnostic processing of all the modules in the rack and reports to the SDP. It also provides local storage of the diagnostic processing results that are beyond the memory capacity of the diagnostic system local to each module.

The module diagnostic processor (MDP) performs module-level diagnostic processing, including built-in self-test for each of the very high speed integrated circuit (VHSIC) chips in the module. It also performs continuous environmental stress measurement,

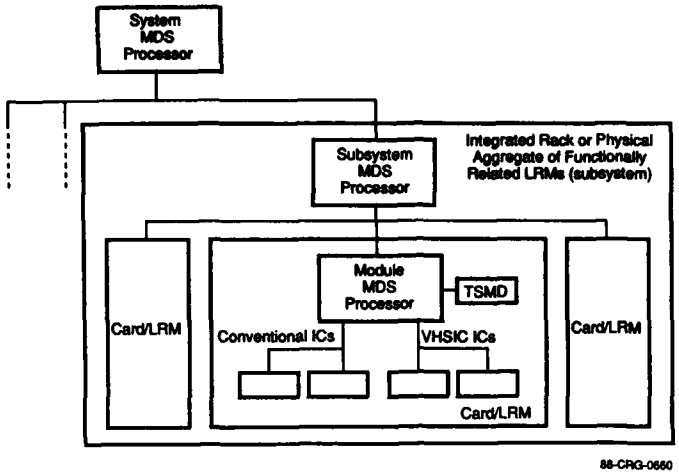



**Figure 3. An Embedded Maintenance Diagnostic Hierarchy**

collecting data on relevant environmental parameters such as temperature, vibration, shock, corrosion, humidity, and g-force. Information of this type is captured through a time-stress measurement device (TSMD) and is reported upon failure or request. Each MDP supplies its module status to the SSDP for the rack.

Results of trade-off studies on lifecycle-cost, power, weight, and mission objectives will determine the particular requirements for redundancy and the techniques for implementation. Whatever redundancy management scheme is adopted will necessarily become part of the integrated diagnostics approach. Resources of the implementation (e.g., self-checking processor pairs, graceful degradation, sensor reconfiguration) must be factored into the support structure.

### Off-Aircraft

Although the technical decisions of the aforementioned programs aid in an integrated diagnostics approach, the philosophy of common modules and standardization also assists by restricting the use of vendor-supplied unique components during implementation. This in turn reduces the amount of unique support equipment, spares, and personnel required to support the system, thus providing a means for controlling support costs while increasing mobility and availability.

## Constraints

Without a totally open design environment there will always be some constraints which limit the scope or impede the application of a true integrated diagnostics approach. With respect to avionic systems, these constraints arise in the areas of interfaces, retrofit approaches, and time pressures.

The integrity of flight-critical functions must be preserved regardless of the interactions between the functions and the support structure. An ideal situation for a fault-tolerant system would include the preservation of these critical functions as well as providing to the diagnostic system the information gleaned by the fault-tolerant functions (e.g., redundancy management and reconfiguration functions). Unfortunately, this increases the complexity of the required interface between the flight-critical system and the diagnostic system, and violates the "keep it simple" guideline for flight-critical functions. The need to increase the complexity of the interface also runs contrary to the desire to maximally isolate the flight-critical system from the possibility of external effects.

The combination of these two circumstances, along with the sad but often true observation that maintenance support is often one of the last implemented functions, means that data are often available to the flight-critical functions that are not obtainable by the diagnostic process during normal operating modes. The converse is understandably true, since use of the diagnostic results by the flight critical functions (e.g., to support reconfiguration decisions) would cause the diagnostic functions to become flight-critical as well, seriously elevating costs.

*Justification for retrofitting an integrated diagnostics concept to an existing operational system can only be arrived at through an availability/cost trade and may require unique operational considerations.* For example, it is easy to imagine conditions where practices may vary between operational sites and depend quite extensively on personnel expertise and daily experiences. It is clear, though, that to achieve true integration, some effective retrofit is necessary in the majority of cases. If retrofit is not implemented, another special case must be tolerated.

Another obvious constraint is the time pressures that are imposed on an operational unit. In an effort to achieve aircraft turn time in support of a particular sortie rate, the degree to which diagnostics are accomplished may vary significantly. Notably, functions that might otherwise be postponed to on-ground performance are instead performed in-flight in an effort to reduce turn time.

## Approach

The above discussion suggests a need to have the various players in the problem solution communicate effectively. Thus, the approach discussed here will address the means to enhance communication: the common language that can be used to allow the elements to work together.

The core of our approach emphasizes identifying and exploiting the similarities between various elements of the integrated diagnostics domain. In particular, the (often overlooked) central focus of all these elements is the physical weapons system being diagnosed. This focus is often lost in the quagmire of technical orders (TOs), TO updates, specialized test equipment, the support equipment for the support equipment (SOS), interactions with forms and databases such as the Standard Base Supply System (SBSS), and so on. Given this common starting point and an identified method of exploiting the common basis, integrated diagnostics approaches realization.

### Integration: What Is It?

Having an integrated system clearly requires more than just declaring that the parts now constitute a whole. From a systems design perspective, achieving integration amounts to:

- Recognizing and enumerating the requirements
- Identifying the constituent elements that satisfy those requirements
- Allotting the functionality across elements
- Defining interfaces between elements
- Reviewing the design to ensure that the requirements are satisfied

The same process holds true for the design of an organizational system. Here, we will focus on the interface definition phase of the integrated diagnostics scenario.

In order to have an effective interface, the information passed must be interpreted readily by the receiving element. If this is not the case, particularly in an organization, the information may be dismissed as too costly to interpret, and the receiving element may choose to regenerate the data (if indeed that is even possible). Part of this ease-of-interpretation (interface) question addresses the mass of data that must be examined. This is less of a constraint when the examination is performed by computer than when performed by a human. Thus the ability to pass information in a highly structured, computer-interpretable form is desirable. An obvious corollary is that the structure used must be common across the generating and receiving elements, and that the information passed must be consistent (or nearly so) with the information already in place in the receiving element.

### Data Sharing

It seems clear that if the various elements of the integrated diagnostics domain are to achieve the goals outlined earlier, there must be a means of sharing data to minimize replication of effort. Having such a data-sharing mechanism requires:

- Identifying the underlying structure of the data
- Identifying a reasonable source for the needed data
- Defining a means of translating the source data into the forms you really require
- Ensuring that the data you have are valid for the intended application

#### Type

All diagnosis can be said to be based on some model of the system being diagnosed. That model may be represented in various ways, for example, in terms of:

- An analytical model (in a simulation system)
- Rules of thumb (in a rule-based expert system)
- Analogy (in the mind of a technician experienced on some similar system)
- A set of objects representing elements of the system (in an object-oriented programming paradigm)

We have chosen the last representation scheme, believing that the model is more clearly related to the physical system.

A model-based diagnostic system is based primarily on design knowledge rather than technician expertise. Model-based expert systems were formulated in an attempt to capture how the expert diagnostician thought about the problem rather than the results of his thinking. It was found that the expert would trace information flow through the system to arrive at suspect components and then run tests to isolate faults among the suspects. The technician expert relied heavily on the system schematics to provide information flow from which he deduced the suspect list. For instance, if the aircraft surface did not respond to a pitch trim command, the expert would identify all the links between the measurement point (the surface) and the stimulus point (the pitch thumbwheel).

A diagnostic model is the cornerstone of the technician's approach. The expert's diagnostic model is a representation of the system that includes his personal knowledge and perceptions of how the system works along with the schematics and descriptions of the physical components. As the technician's internal model more closely approximates the physical system, his diagnosis of any given problem is quicker and more accurate.

In the case of a model-based expert system, the diagnostic model is also the cornerstone. If the information flow or functional connectivity of each line-replaceable unit (LRU) can be captured within the computer, then the computer can make judgments as to which elements lie along a fault path. Thus the knowledge stored in the model is not just a symptom-to-fault mapping, but is a mechanism to functionally trace which components could be responsible for a given observation. Thus, the MDS (computer, technician, and model) combines a very detailed model of the system under test—which is stored in the computer memory—with the intuition of the maintenance technician. This allows the technician to focus his attention on the diagnostic tasks.

The fault isolation guide was created using these same information-flow techniques. The expert hypothesized faults and then built fault trees that guide the technician to obtain information about the health of each of the LRUs in the connectivity path between stimulus and measurement. A model-based approach to diagnostics eliminates the need for fault trees. This is accomplished by the computer dynamically choosing which test provides the most information for the least amount of effort and time expended by the technician at each step in the fault diagnosis process.

In one system that is representative of this approach, the Flight Control Maintenance Diagnostic System (FCMDS), the diagnostic model consists of a set of objects including the LRUs, subLRUs (or functional elements), signals, cabling, access doors, switches, and so forth. The available test actions are represented as another set of objects, and a small algorithm selects an appropriate test action, updates the physical system model, and again selects a test action (Reference 1).

### Source

In order to ensure that the model we construct is indeed relevant or true to the system being diagnosed, we rely on source data documenting the system under diagnosis. In some cases, this means reliance on the paper technical data delivered with the system, which are often imperfect. For most current fielded systems these data are not machine readable and hence a human translation effort is required to obtain the desired model. The human translation can introduce another set of errors. The on-line representation of these technical data is little better as a source for automated translation, since the representation is aimed at satisfying publishing requirements, not model generation.

In future systems, computer-aided design (CAD) data will form a far better source for model generation, particularly when the CAD representation is used for simulation (or other uses) where a "deep" model is needed. The Computer-Aided Acquisition and Logistics Support (CALS) Initiative and the consequent increased emphasis on Initial Graphics Exchange Standard (IGES) and Product Data Exchange Specification (PDES) make the likelihood of having data available for this purpose far greater.

### Processing

The end result of the data processing to produce the diagnostic models we use is essentially fixed. Thus the complexity of the processing involved, and the associated likelihood of introducing errors, is proportional to the "length" of the translation process. In severe cases (paper documentation), the translation process requires the intervention of a human, which makes the translation *process* unverifiable (though not the results). Systems which are being designed today are almost universally available as CAD representations (see Figure 4).

The result of this translation process from the CAD source is an extremely detailed model of the system. The detail of this representation is suitable for use at the depot level, for example, but has too fine a grain for use in on-aircraft or in-flight applications. For these applications, the model can be "lumped" by examining the testability of the system and reducing fidelity to the representable elements for the available test set at a given organizational level.

### Validation

For the diagnostic system to be accepted, its use must be known to be free of errors which would introduce safety hazards for either the maintenance crew or the crew of the weapons system. The diagnostic system's performance must also be correct in the sense that existing faults will be found and corrected, and few if any false alarms will be generated.

Expert systems used in diagnosis are inherently more flexible than the TOs they are designed to augment or replace. This very flexibility makes them hard to verify, since the range of possible behaviors is so broad. The model-based approach reduces the complexity of this verification task by isolating the procedural (algorithmic) part of the diagnostic system from the declarative (model facts) part. If the behavior of the relatively small algorithmic segment can be verified, then its actions on any correct model are well defined. The model's correctness, in turn, depends on correct source data and a correct translation design and implementation. For system designs that are available in CAD formats, the source data are verifiable by means of simulation, consistency checks on the database, and the like. The usual approach to verify the correctness of the translation is to perform a reverse translation and compare the original to the twice-translated form. This twice-translated verification approach has been used extensively, for example, in the U.S. Navy's latest submarine design effort.

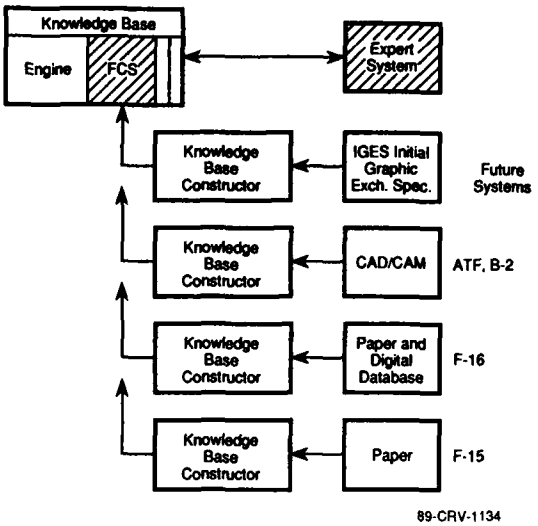



**Figure 4. Information Production Advancement**

## Replication

There is a common data source for each of the levels of granularity of the diagnostic model that is eventually used in in-flight, on-aircraft, and off-aircraft maintenance/diagnostic scenarios (Figure 5).

If the generation of these models is a manual effort, then this can be viewed as a replication of effort. Even if one makes this argument, it is no worse than the current state of affairs in which the separate organizational levels have technical documentation specific to each level. In fact, it is arguably better, since the creator of a new model has previous models on which to base his effort. This granularity is consistent with the "cone" of limits associated with levels of testing.

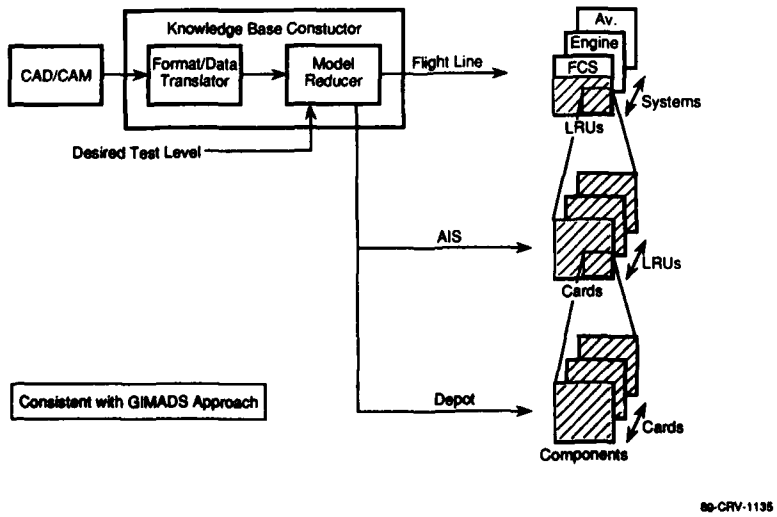



**Figure 5. Common Data Across Organizational Levels**

In the automated scenario, of course, the situation is far better. The generation of the most detailed model is a straightforward process which involves:

- Identifying the constructs you need to know in the target model
- Identifying the source of those constructs in the source model
- Identifying modifications to the target model to support constructs existing in the source but not the target
- Implementing the target model modifications
- Designing the mapping from source to target
- Performing a review with people familiar with source and target
- Implementing the mapping
- Testing (through reverse translation, implemented in a similar fashion by a separate team)

## Implementing Commonality

As we have said, the commonality that we can exploit with the functional model-based approach centers around the fact that all the models come from the same source. In fact, we are now exploring techniques which optimally reduce the full model (perhaps automatically generated from CAD inputs) to one which has sufficient resolution to encode and interpret the available inputs, *and no more*.

In this way, the requirements on memory and processor are effectively constrained to be (at a maximum) those consistent with the type of information processing that can be done given the information available. This process is shown in Figure 5.

## In-Flight

### Model

The complexity of the model for the in-flight case is far less than for other cases. In some instances, it will be limited to something more simple than what could be supported given the available BIT capability and redundancy management information, due to the lack of processor and memory spare capability. For newer systems, this is less of a problem, and with the advent of distributed architectures that support the local processing of diagnostic information and the communication of raw data and results between elements, the capability for a diagnostic element to store information and diagnose over a small set of functional elements becomes realizable. This is particularly attractive in the case of architectures having a small number of distinguished types of processing elements, such as the PAVE PILLAR approach. In this case, the diagnostic model (though not the state of the computation) is replicated, and can be shared across chassis if a given diagnostic processor is disabled or must be allocated to a higher criticality function.

### Process

The in-flight process is determined by the requirements imposed on the performance of the system. The minimum capability is simply to record raw data for later processing. A more advanced capability performs in-flight reduction of the data, makes rudimentary decisions about the usefulness of BIT samples to the ground-based system, then compresses and time-stamps the data it chooses to keep.

The next level of processing modifies the sampling performed in-flight when the diagnostic system determines that something "interesting" is happening. This may be implemented as a circular queue whose contents are dumped as a "window" around a given BIT indication or time-stress threshold crossing. A more advanced sampling implementation may choose to run initiated BIT on a system having intermittent failure indications to isolate the fault while it is active. To date, this is the only identified approach to reducing the cannot-duplicate problem, namely to collect information about the context in which the failure occurred.

The most advanced in-flight processing capability uses the status updates to adjust the diagnostic systems confidence in the operational status of monitored systems. In a PAVE PILLAR architecture, this information might be used to influence the allocation strategy for pending processes to elements believed more reliable, though this sort of approach has implications about the flight criticality of the diagnostic system itself.

### Results

The type of output or results that are available from in-flight processing is primarily determined by the data made available to the diagnostic system and the sophistication of the processing that occurs. Even in older systems such as the F-16A (Block 10), enough data are available within the electronic component assembly (ECA) and flight control computer (FLCC) to reduce the ambiguity group to 2 or 3 in many cases, though no attempt is made to process this data in-flight. The realization that subsystem designers often provided BIT capability (often for the purpose of supporting redundancy management) that was not available to external users under normal operational conditions has led to the definition in the PAVE PILLAR world of the Test and Maintenance Bus, a communications path dedicated to this type of traffic. The availability of data, and the intent to provide spare resources that can effectively be used to diagnose problems in-flight, yields the ability to determine the needed support level (austere vs. full) and, if desired, to radio ahead to prepare the appropriate equipment and spares to speed the turn.

## On-Aircraft

### Model

An example of the *on-aircraft model* exists as part of the FCMDS. Here, the model was written by hand using the Air Force Technical Data as the source. It is important to note that there are two "versions" of this model: the one used for development and authoring, and the one used in the portable system, which is automatically generated from the development version by stripping out the redundant and "human oriented" data. The model represents the functions of the F-16A (Block 10) flight control system, the signals which support those functions, the LRUs that contain the functions, the access panels, switches, cabling runs, etc. needed to support guiding the maintenance technician through the tests that FCMDS suggests.

### Process

In the current FCMDS system, the technician is first instructed to run a set of operational checks to verify failure (see Figure 6). Self-test is then begun, with FCMDS prompting the technician to relate any failed steps, and the associated indications. Generally, at the end of self-test, the LRU ambiguity group is larger than 1. (If this is not the case, FCMDS recommends a remove and replace action, and retests the system.) FCMDS then screens the available set of tests according to whether they check a functional element that FCMDS currently suspects. The remaining tests are prioritized based on a cost/benefit analysis merged with a divide-and-conquer strategy. The technician is guided through the procedure, and the test results are analyzed. The process is repeated until all the suspect functional elements lie within one LRU. (This is an abbreviated process description; for a more complete statement, see Reference 2.)

### Results

FCMDS will guide the technician through test procedures until either the set of all suspect functional elements lies within one LRU, or there are no remaining untried test procedures. We have recently field-tested the FCMDS system at McDill AFB, and more extensive field test efforts have recently been funded at McDill, Luke, and Hill AFBs. The subjective rating score for the initial field test of the FCMDS system was 4.1 on a scale of 5.

## Off-Aircraft

### Model

The model for off-aircraft maintenance is similar in form to that used on-aircraft, though it is substantially more detailed, since the set of applicable tests is so much larger. The model used here is segmented by LRU, since the system interconnectivity which is a concern in on-aircraft maintenance is no longer in place.

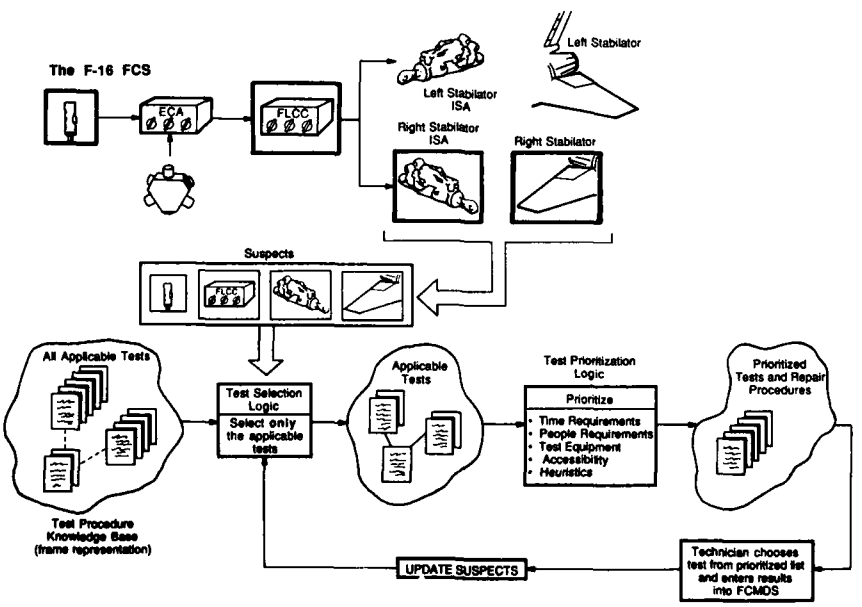



**Figure 6. FCMDS Diagnostic Process Flow**

**Process**

The results obtained in on-aircraft maintenance are available as inputs to the off-aircraft procedure. Since the on-aircraft procedure collects information about the functional elements which are suspect, the off-aircraft maintenance personnel have clear indications as to which parts of the model are potentially failed, hence which tests are immediately applicable. This information can be used to prioritize test sets in test equipment which supports test set reordering, and to generate an applicable test set in future systems.

**Results**

The failed component is readily identified, and this information is tied to the model for collection and analysis at the fleet level for reliability enhancement. Redesign recommendations are associated with the faulty model segment; this aids in unambiguous identification of the problem as well as model update verification after redesign (the model regenerated from the new CAD file is compared to the old model and the changed areas are determined to be in identified segments).

## Conclusions

The functional model-based diagnostic approach provides a common basis for information transfer between various elements of the integrated diagnostics scenario, minimizing the translation necessary when passing information from one organizational level to another. The functional model can be generated in a more straightforward manner than other diagnostic representations, leading to cost-effective implementation. In the future, on-line structured representations will lead to automatic generation of a highly detailed base model from which less detailed models specific to an organizational level may be developed.

## References

1. Bursch, P.M., Meisner, J.W., McAfoos, R., and Schroeder, J.B., "The Flight Control Maintenance Diagnostic System," *NAECON-88*, Dayton, Ohio (1988), 1504-1509.
2. Bursch, P.M., Meisner, J.W., and Winegar, K.F., "A PC Based Expert Diagnostic Tool," *AUTOTESTCON-87*.

# A BYZANTINE RESILIENT PROCESSOR WITH AN ENCODED FAULT-TOLERANT SHARED MEMORY

by

**Bryan Butler** and **Richard Harper**
Fault-Tolerant Systems Division
The Charles Stark Draper Laboratory
Mail Stop 3E
555 Technology Square
Cambridge MA 02139
United States

## Abstract

The memory requirements for ultra-reliable computers are expected to increase due to future increases in mission functionality and operating-system requirements. This increase will have a negative effect on the reliability and cost of the system. Increased memory size will also reduce the ability to reintegrate a channel after a transient fault, since the time required to reintegrate a channel in a conventional fault-tolerant processor is dominated by memory realignment time.

In this paper, a Byzantine Resilient Fault-Tolerant Processor with Fault-Tolerant Shared Memory (FTP/FTSM) is presented as a solution to these problems. The FTSM uses an encoded memory system, which reduces the memory requirement by one-half compared to a conventional quad-FTP design. This increases the reliability and decreases the cost of the system. The realignment problem is also addressed by the FTSM. Because any single error is corrected upon a read from the FTSM, a faulty channel's corrupted memory does not need realignment before reintegration of the faulty channel. A combination of correct-on-access and background scrubbing is proposed to prevent the accumulation of transient errors in the memory. With a hardware-implemented scrubber, the scrubbing cycle time, and therefore the memory fault latency, can be upper-bounded at a small value. This technique increases the reliability of the memory system and facilitates validation of its reliability model.

## 1. Problem Statement

The memory requirements of ultra-reliable computers are bound to increase due to increasing mission-critical functionality and the needs of memory-hungry languages such as Ada. As the memory size increases, the probability of computer loss, which is dominated by memory, increases commensurately. The memory is expensive as well; the ESA's Ulysses spacecraft recently replaced 6Kbytes of RAM at a cost of ~$1M for hardware alone [1].

In a redundant computer, reintegration of a channel which has undergone a transient fault requires alignment of that channel's memory to the same state as that of the correct channels. Typically, the memory realignment *must* be completed before the application task can be resumed. The time required to perform this alignment may be several seconds for a large memory, making recovery infeasible for fast real-time applications. This is exacerbated by the fact that transient faults occur from ten to one hundred times more frequently than permanent faults, depending on the operational environment.

To prevent excessive accumulation of latent soft errors in the memory, a memory scrubbing task is often used to periodically read, vote, and write back the corrected contents of each location. Because this task consumes processor throughput, it is typically run in the background. In a computationally loaded system, the time required to cycle through a large (e.g., 1 Mbyte) memory may be on the order of an hour. During this time, latent errors may accumulate in one or more channels, exposing the system to loss due to near-coincident error manifestations in more than one channel. Moreover, it is difficult to upper-bound the memory error latency since it may depend on computational load, making validation of this particular aspect of the system's reliability model a difficult task.

## 2. Problem Solution Approach

Encoded memories have been proposed to solve some of these problems. In previous work, Krol [2] describes a memory system using a (4,2) linear separable code for informational redundancy. Each channel possesses a processor, a local symbol memory, an encoder with which it generates a symbol to write into its own memory channel, and a decoder with which it generates a decoded output from the symbols emanating from all channels upon a read. Although not explicitly stated by Krol, the four symbol memories can be arranged into the four fault containment regions comprising a processing site which meets the requirements for Byzantine Resilience [12]. However, because the (4,2)-FTP as presented in [2] does not perform the source congruency function on channel-specific or single-source data, it is not clearly Byzantine resilient.

Because such a memory requires one half of the memory chips of a quadruplicated design (e.g., quad-FTP [4], SIFT [5], NEFTP [6], MAFT [7]), the failure rate $\lambda_m$ of the encoded memory system is approximately one-half that of a quadruplicated design. Since short-term system loss probability is proportional to $\lambda_m^2$ [3], the computer's system loss probability is quartered. The Mean Time to Failure is approximately doubled, since it is proportional to $1/\lambda_m$.

An encoded memory using the (4,2) code can correct any single arbitrary symbol error. Since the decoding circuitry will correct a corrupted symbol upon a read, reintegration of a channel possessing corrupted memory does not require immediate memory alignment. The corrected contents will be written into the corrupted channel upon a symbol write. Moreover, alignment of the faulted channel's memory is accomplished in the normal course

of memory scrubbing. Finally, a hardware-implemented scrubber can be used to sequentially read the memory through the decoder, correct any errors, and write the corrected symbols back to the memory on a periodic cycle-stealing basis, thus upper-bounding the memory fault latency to a smaller value than that provided by a processor-based scrub scheme.

This paper describes the use of an encoded memory-based fault-tolerant processor architecture (denoted the Fault-Tolerant Processor with Fault-Tolerant Shared Memory, or FTP/FTSM) under development at The Charles Stark Draper Laboratory to address these problems. First, an overview of the architecture and its operation are presented. Next appears a reliability analysis of the FTP/FTSM, where it is compared to quadruply redundant designs. Concluding the paper is a performance analysis of the FTP/FTSM, which relates the effect of memory read and write overhead to throughput.

## 3. Theoretical Requirements for Byzantine Resilience

The primary objective of a fault-tolerant computer is to survive faults by containing and isolating their effects. A failure of one component should not cause failures in other components. Two approaches have been taken to achieve this goal. The first approach is to enumerate and estimate the likelihood of failure modes for each component. Ad-hoc fault-tolerance techniques are then developed for each hypothesized failure mode such that the more likely failure modes will not precipitate faults in adjacent components. However, the enormous complexity of computer systems and fallible human bias make this approach extremely difficult, expensive, and of doubtful effectiveness for ultra-reliable systems.

A more universal approach can be taken if no assumptions whatever are made about possible failure modes. This is the approach known as Byzantine resilience. Fault isolation is obtained by physical and electrical isolation of groups of components into fault containment regions (FCRs), also known as channels or lanes. The failure of one component within one FCR may cause the failure of other components within that FCR, but cannot induce faults in another FCR. Moreover, arbitrary behavior in one FCR cannot cause the aggregate of FCRs to exhibit erroneous behavior.

Error propagation occurs when a faulty FCR emits corrupted data to another FCR. If a functional recipient FCR does not react the same as other functioning FCRs, that FCR may appear faulty. Informational redundancy and a fault masking function are used to prevent corrupt data from degrading one or more functioning FCRs. Redundant information is delivered to an FCR from other FCRs. The recipient FCR applies the fault masking function to the redundant data, thereby masking a given number of erroneous data items.

The theoretical requirements for Byzantine resilience have been demonstrated in a number of studies ([8], [9], [10], and [11].) An F-Byzantine resilient system, able to tolerate the simultaneous loss of F fault containment regions, must meet the following requirements:

- 3 F + 1 fault containment regions are required (cardinality requirement)
- Each FCR must be connected to at least 2 F + 1 other FCRs by disjoint communication links (connectivity requirement)
- F + 1 rounds of exchange are required to distribute single-source data (source congruency)
- The functioning FCRs must be synchronized to within a known skew (synchronization requirement).

## 4. Error-Correcting Codes

The FTSM uses an encoding scheme, known as the (4,2) code, to encode data words before they are stored into the RAM of the FTSM. The (4,2) code is the same code described in [2] for use in the (4,2)-concept FTP. The encoding process takes two 4-bit symbols which represent a data byte and generates four 4-bit symbols. The code is designed to tolerate any single symbol loss. Each FCR stores a different symbol so that if one FCR is lost, the data word can be reconstructed from the three remaining symbols.

The generation of the (4,2) code begins with the definition of a Galois Field of $2^4$, or 16, elements. This definition includes an addition and a multiplication operation, both of which are closed over GF($2^4$). The addition operation defines an abelian group, and the multiplication operation defines a commutative monoid (every element except zero has a multiplicative inverse). The details of the (4,2) code are presented in [2] and [12].

The code is designed to tolerate a number of error conditions. In one mode of operation, known as the random mode, the code can tolerate the loss of any symbol (from one to four bits in error) or the loss of any two bits (in the same or different symbols). Another mode, known as erasure mode, allows an additional bit error to be corrected in the presence of a known symbol in error. In this mode, one of the four symbols is suspected of being wrong. Note that it is not necessary to know the value of the error; only the position of the error must be known. A third mode, known as duplex mode, allows the extraction of data when two symbols are suspect. The duplex mode is incapable of masking any errors beyond the two suspect symbols.

## 5. Use of (4,2) Code as a Fault Masking Function

A fault-tolerant computer requires a fault masking function so that, in the presence of faults, an FCR can resolve redundant information from multiple external FCRs into a single value. The fault masking function should be deterministic, i.e. separate instances (on different FCRs) should return the same result given that the input values are identical.

The traditional fault masking function uses nominally identical copies of data which are resolved using a bitwise majority vote, shown in Figure 1. Each bit position is voted independently of all other bit positions. To perform a majority vote, three (or four, in the case of a quad-FTP) copies of data are kept. Each copy is stored in a separate FCR so that if one copy is lost, the remaining copies will constitute a majority from which the data can be recovered. This scheme requires total storage on the order of 3n or 4n and inter-FCR communication bandwidth of order n, where n is the width of the data.

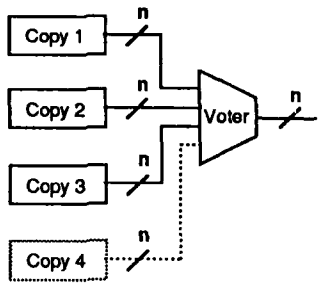



**Figure 1: Majority Vote Fault Masking Function**

The (4,2) code presented above can be used as a fault masking function in place of the traditional bitwise majority vote function found in most fault-tolerant computers. The (4,2) code is a four-symbol minimum-distance separable (MDS) error correcting code. An encoder generates, from a data word, four symbols which provide informational redundancy for the data word. Each symbol is stored in a different FCR. Fault masking is performed by a decoder. The decoder uses the four symbols to regenerate the data word. If one of the symbols is corrupted, the decoder will still be able to recover the data from the three remaining symbols.

The use of the (4,2) encoding scheme as a fault masking function is demonstrated in Figure 2. A data word of size n can be stored as four symbols, each of size n/2, with only three symbols needed to recover the data in the presence of a single random error. The resulting storage requirements are of order 2n, and the communication bandwidth is of order n/2.

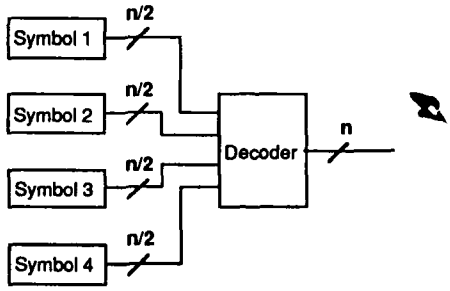



**Figure 2: (4,2) Code Fault Masking Function**

While the effect of the (4,2) code is similar to the majority vote function, a few subtle differences exist which must be thoroughly investigated to ensure that the FTP/FTSM design is Byzantine resilient.

The majority vote function requires at least three redundant copies as inputs to determine an unambiguous output in the presence of a single error. More than three copies can be used, as long as only one is assumed to be in error. The (4,2) code requires exactly four symbols as inputs, of which one symbol can be in error. Another difference between the two functions is that each copy in a voted system contains all information necessary to reconstruct the data. If any given copy is determined to be correct, the data value is readily available. For the encoded system, at least two symbols are required to recover the value of the data object.

The consequence of these differences is that the interchannel information exchange operations are not necessarily the same as those for a traditional quad-FTP. The exchange operation to resolve commonly sourced output is similar, with symbols replacing copies. However, source congruency must be done differently. Byzantine resilient agreement or validity (whichever is appropriate) can be assured if either of the following two conditions is met:

- All functioning FCRs agree on the value of three of the four symbols, and these three symbols map to a valid code word.
- All functioning FCRs agree on the value of all four symbols.

A traditional source congruency exchange substituting symbols for data copies will achieve the first condition if a recipient FCR is faulty. The second condition must be guaranteed to account for situations where the source FCR is faulty. The problem is that, if the source FCR is faulty, some of the recipients may have a set of symbols which maps to a valid code word, whereas others do not. This situation is intolerable, since all functioning FCRs must make the same decision. The problem can be solved by performing a second source congruency with a majority vote on the symbol emanating from the source FCR. While the source congruency operation for the FTP/FTSM requires more exchange cycles and is more complicated than a source congruency for a quad-FTP, the resulting exchange primitive is still more efficient than a quad-FTP [12].

## 6. Architecture Overview

The design presented below is a fault-tolerant processor which uses a fault-tolerant shared memory system. To the processing elements, the fault-tolerant shared memory (FTSM) appears to be a simplex, highly reliable shared memory to which all processing elements have access. In reality, the FTSM is distributed among the fault containment regions with physical and electrical isolation between the FCRs to eliminate all single-point failure modes. The FTSM is designed to meet all of the requirements for Byzantine resilience.

The FTP/FTSM is a computer system which makes use of a fault-tolerant shared memory (FTSM) system to enhance reliability. The FTSM is partitioned into pieces, called quadrants, such that the loss of any quadrant will not cause the loss of data or the functionality of the computer. This configuration is shown in Figure 3. Each quadrant of the FTSM is contained in a separate FCR. If one quadrant of the FTSM is lost, the remaining three quadrants contain enough information to reconstruct all data values.

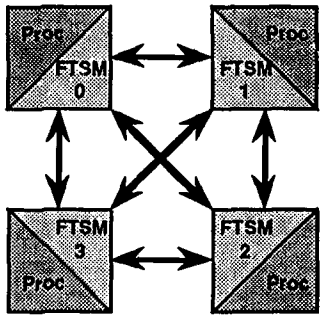



**Figure 3:Physical Configuration of FTP/FTSM**

The FTSM meets all of the requirements for single-fault-Byzantine-resilience for the following reasons. Each FTSM quadrant, of which only three are required for functionality, is contained by a separate FCR to conform to the cardinality requirement. The connectivity requirement is satisfied by an inter-FCR communication system (IFC) which connects every quadrant to all other quadrants. The source congruency operation is a function performed by the FTSM at the processors' request. A fault-tolerant clock (FTC) is distributed by the IFC to perform FCR synchronization at the micro-instruction level.

The FTSM system can be viewed as a Byzantine resilient shared memory system to which four processors are connected. This view, shown in Figure 4, is the programming model of the FTP/FTSM. Since each processor is connected to the FTSM, all processors can read a value from any location in memory. Whenever the processors perform a memory read cycle, the FTSM quadrants exchange the code symbols corresponding to the requested location. Each quadrant receives three symbols from the remote quadrants in addition to the locally stored symbol. The coding scheme used by the FTSM is designed so that a symbol from one quadrant can be arbitrarily corrupted without corrupting the resulting data. The decoder in the FTSM quadrant uses the four symbols to regenerate the original data word. Note that, unlike most other fault-tolerant computers, the data exchange and fault masking is performed implicitly upon memory reads. This implicit fault masking is what gives the FTP/FTSM the ability to recover from transient faults before realigning memory.

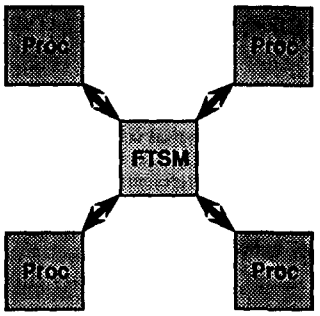



**Figure 4: Virtual Configuration of FTP/FTSM**

## 7. Reliability Analysis of FTP/FTSM

The purpose of the reliability study is to compare the reliability of the FTP/FTSM with other systems and quantify the effects of the design decisions. A complete description of the parameters and analysis procedures used to obtain the various analysis results is presented in [12].

### 7.1. FTSM Encoded vs. Quadruplicated Memory

Figure 5 shows a comparison of the system loss probability for the FTSM, which uses an encoding function for fault masking, and that of a similar shared memory system using a quadruplicated memory with a majority vote function for fault masking. The same Markov model was used to model both systems, the only difference being the number of bits per symbol/copy. All parameters relating to the comparison are the same, including read rate, write rate, scrub rate (per location), and bit failure rate. In the FTP/FTSM, the encoding scheme reduces the memory complement of the processor, and therefore its failure rate, by a factor of 2. The effect of this reduction on system loss probability is shown in Figure 5. Table 1 presents the per-hour system failure rates due to transient and permanent faults, illustrating that the FTP/FTSM's failure rate is reduced by a factor of four over the quadruplicated design.

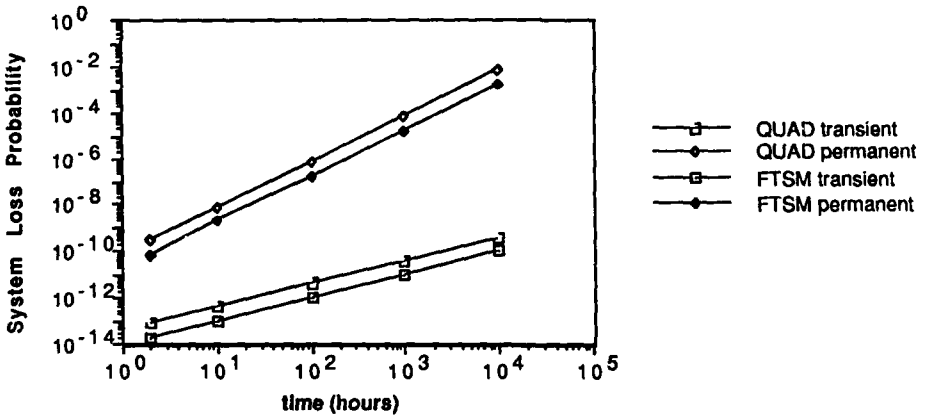



**Figure 5: System Loss Probability of Encoded vs. Quad Memory**

| Architecture | Due to Transients | Due to Permanents |
|---|---|---|
| FTP/FTSM | 1.03 e-14/hour | 3.73 e-11/hour |
| Quadruplicated FTSM | 4.10 e-14/hour | 1.49 e-10/hour |

**Table 1: System Failure Rate of Encoded vs. Quad Memory**

This analysis also demonstrates that transient errors are much less likely to cause system loss than permanent errors, even at short mission times. This characteristic is caused by the memory scrubber. The scrubber can correct transient errors if they are caught early enough so that an uncorrectable condition does not occur. If the

scrubbing operation is performed at a much higher rate than the transient error rate, the effect of transient errors on memory system reliability is effectively eliminated.

### 7.2. Reliability vs. Scrub Rate

Because of program locality of reference, most locations in a memory system are accessed very infrequently over short-periods of operation. A similar skewed workload distribution is also expected for long-term operation. Previous studies indicate that the workload distribution can have an effect on the reliability of a self-checking memory system [13]. Memory access rates are highly non-uniform across the address space, with a few locations receiving many accesses and most locations being referenced very rarely. The graph in Figure 6 shows how the access rate distribution might look for a real system, and a simple modeling estimate for the distribution. The value f indicates the fraction of "fast" memory locations which have the high access rate. Note that f will probably be small (< 1%) for systems with a large amount of memory.

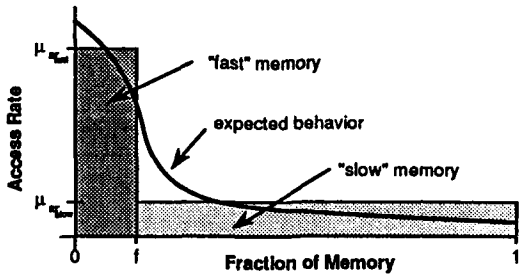



**Figure 6: Memory Access Distribution**

The graph in Figure 7 shows the reliability of the FTSM memory system (without scrubbing) as a function of the percentage of so-called "fast" memory. This graph demonstrates that the reliability of the memory system is bounded by the reliability of the "slow" memory for all but unreasonably high percentages of "fast" memory. To improve the reliability of the FTSM system, the reliability of the "slow" memory must be increased. The memory scrubber accomplishes this task by periodically accessing every location in memory at a fixed rate, placing a lower-bound on the access rate, and thereby the error latency, of the "slow" memory.

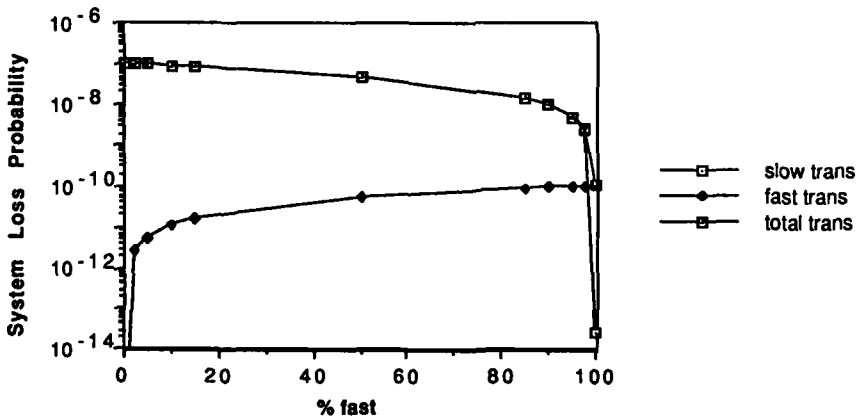



**Figure 7: System Loss Probability vs. % Fast Memory**

The effect of scrubbing rate on system reliability is shown in Figure 8 for an FTSM with 0% "fast" memory. The "fast" memory was ignored since it has little effect on reliability (see [12] for actual access rates). Both transient and permanent system loss probabilities are shown for missions of 10, 100, and 10000 hours in duration. Scrubbing has little effect on permanent errors, other than their early detection, so the permanent system loss probability is nearly constant with respect to scrubbing rate. The "knee" in the transient system loss probability graph occurs when the scrubbing rate is equal to the memory access (write) rate. If the write rate is higher than the scrubbing rate, the scrubbing operation has little effect on the reliability, whereas a higher scrubbing rate dominates the reliability otherwise.

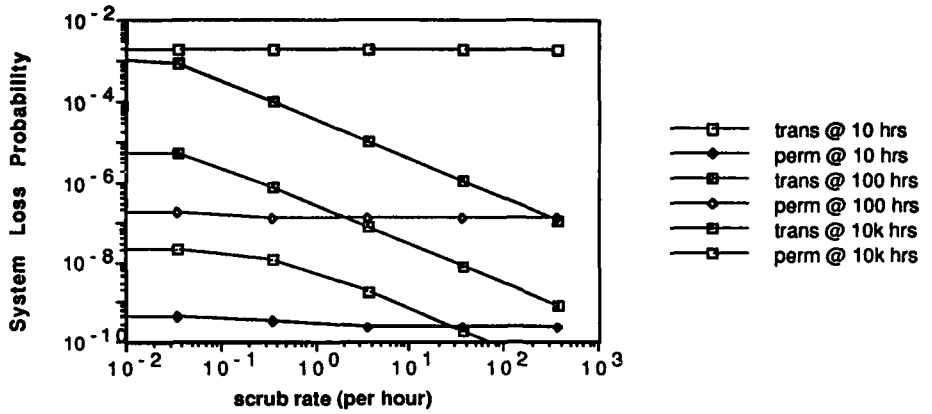



**Figure 8: System Loss Probability vs. Scrub Rate**

Several interesting conclusions can be drawn from this graph. First, scrubbing is far more important for increasing short-term system reliability than for long-term reliability. This is easily understood since transient errors, which are correctable by the scrubber, dominate short-term reliability, whereas uncorrectable permanent errors dominate long-term reliability. Also, relatively slow scrub rates yield a measurable increase in short-term reliability. At 10 hours, the system loss probability can be reduced to the limits of permanent failures with a scrub rate of 30 per hour per location.

The required scrub rate may be low enough that the scrubbing function can be implemented in software instead of hardware. The inclusion of the hardware scrubber may be useful in computationally stressed systems where a software scrubber would impose extremely high performance penalties, while the increased hardware complexity may make the hardware scrubber undesirable in other circumstances. The issue of hardware or software implementation is indeterminate without more information about the actual application in which the FTP/FTSM is to be used; once the application has been characterized, the analysis summarized above can be used as the basis for a quantitative engineering trade-off decision.

## 8. Performance Analysis of FTP/FTSM

The primary interest in the performance analysis of fault-tolerant computers is the performance penalty imposed by the fault-tolerant mechanism. One way of analyzing the performance of a fault-tolerant computer is to compare the time required to complete a specific task on a comparable simplex (i.e., non-redundant) processor to the time required to complete the same task on the fault-tolerant processor. Another performance metric is the overhead imposed on a task by the fault-tolerance functions.

A reasonable benchmark for these analyses is an iterative control loop, since FTPs are often used for real-time control systems. A typical application for an imbedded real-time system is flight control. The control loop for this application, shown in Figure 9, consists of three parts. During the first, data from input sensors is read by the host FCR and exchanged via the source congruency exchange. The actuator outputs are computed during the second part. The third phase involves voting the actuator outputs to detect and mask computational faults and transmitting the output values to the actuators.

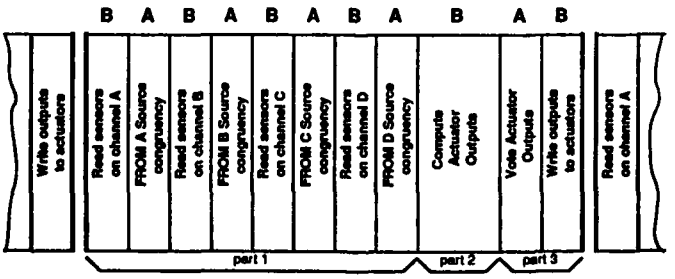



**Figure 9: Control Loop Iteration for Flight Control**

Each step in the control loop shown above can be classified as belonging either to the fault-tolerant task or to the computational task, as indicated by the labels A and B, respectively. Any step which would be executed by a

simplex computer performing the control task is defined to be part of the computational task, even if the step involves non-computational operations, such as I/O. The function of the computational task is unchanged on different architectures. The fault-tolerance task is composed of those steps which only apply to fault-tolerant computers. The fault-tolerance task varies depending on the FTP being analyzed. The fault-tolerant task time is denoted by $T_A$ and the computational task time is defined as $T_B$. The task completion time for a design Y is the sum of the computational task time and the fault-tolerance task time:

$$T_Y = \sum T_{A_Y} + \sum T_{B_Y}$$

The FTP/FTSM analyzed in this section is compared to a simplex baseline design, a CSDL-designed fault-tolerant processor denoted the VLSI FTP, and a CSDL-designed prototype called the Network Element Based Fault-Tolerant Processor (NEFTP). The baseline design only executes the computational task, for which the task completion time is defined as $T_0$.

Direct comparison of competing architectures can be made by comparing the task completion time on any given design Y to the task completion time on the baseline system, using the task completion ratio:

$$TCR = \frac{T_Y}{T_0}$$

Another interesting parameter is the overhead imposed by the fault-tolerance. While overhead is not necessarily a valid metric for direct comparison, it is an indication of how much of the processor power is tied up with fault-tolerance functions. Overhead is defined as the ratio of the completion time of the fault-tolerance task to the overall task completion time:

$$OH = \frac{T_{A_Y}}{T_Y} = \frac{T_{A_Y}}{T_{A_Y} + T_{B_Y}}$$

### 8.1. Baseline System Performance Analysis

The baseline system for this performance comparison is a 25 MHz Motorola 68030 processor with zero wait-state memory. To compare throughput of each of the FTP architectures with the baseline system, a step-by-step translation will be presented which will account for all of the significant differences between the baseline system and the FTP being analyzed.

### 8.2. NEFTP Performance Analysis

The current incarnation of the NEFTP [6] uses 12.5 MHz 68020 processing elements. For this analysis, we will assume the existence of a hypothetical enhanced NEFTP which uses a 25 MHz 68030 as a processing element. The only translation needed to compare the enhanced NEFTP to the baseline system is the addition of the network element, which performs fault-tolerant specific functions such as voting, source congruency, and synchronization.

The control loop as it might be coded on the NEFTP is shown in Figure 10. The NEFTP has three different functions which are performed during one iteration of the control loop. The FROMx source congruency is used to exchange single-source data from input sensors hosted by each processor. The VOTE function is used to vote actuator output values. SYNC is used to ensure processor synchronization. The penalties imposed by executing each of these functions are tabulated in [12]. The times given for FROMx and VOTE are for packet sizes of 240 bytes.

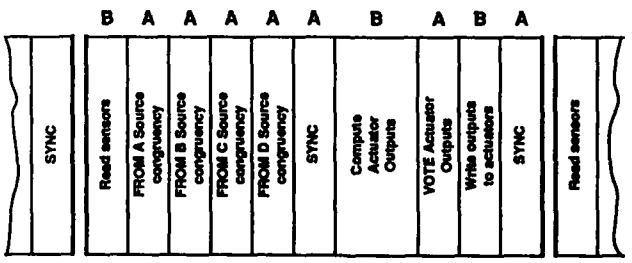



**Figure 10: Control Loop for NEFTP**

The completion time for the computational task is unchanged from the baseline, since the enhanced processing elements are identical to the baseline system. The fault-tolerance task as shown in Figure 10 is composed of four FROMx operations, one VOTE, and two SYNCs. The total time for completion of the fault-tolerance task was measured to be 2.21 ms on the 12.5 MHz 68020. While some of this time is dependent on the processor model and clock rate, in the current analysis we leave this figure unchanged to obtain a conservative estimate of the time required to perform the exchanges on the enhanced NEFTP. Note that for a given packet size and processing element, the fault-tolerance task time is a constant. The total task completion time for the NEFTP is:

$$T_{NSFTP} = T_0 + 2.21ms$$

### 8.3. VLSI-FTP Performance Analysis

A prototype fault-tolerant processor known as the VLSI-FTP was developed at CSDL. This architecture uses the processor/interstage design presented in [4]. The 16 MHz 68020 processing elements are clock deterministic and are tightly synchronized by a fault-tolerant clock. The analysis presented below assumes an enhanced VLSI-FTP architecture which uses processing elements identical to the baseline system.

Translation from the baseline design to the VLSI-FTP requires the addition of the interstage exchange mechanism which implements the necessary interchannel data exchange functions.

The VLSI-FTP implementation of the control loop is shown in Figure 11. The processors in the VLSI-FTP are micro-synchronized using a free-running fault-tolerant clock composed of four phase-locked channels, so no explicit SYNC operation is necessary. However, synchronization does contribute to the fault-tolerance overhead. A fault-tolerant clock is

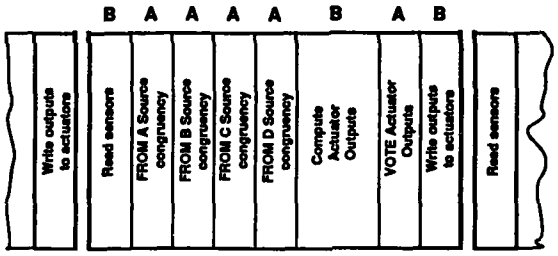



**Figure 11: Control Loop for VLSI-FTP**

nominally 18 cycles of a 25 MHz clock. An average of 1 adjustment to the processor clock reduces the number of processor clocks per FTC cycle to 17. The result is that only 17/18 of the 25 MHz clock cycles are available for processor computation. The cycle that is lost to synchronization needs to be included with the fault-tolerance task completion time, since synchronization is a part of the fault-tolerance functionality. The following equation represents the time required to complete the computational and synchronization tasks:

$$T_B + T_{sync} = \frac{18}{17} T_B$$

The time to complete the computation plus the synchronization is 18/17 the time required to complete only the computation. By solving for $T_{sync}$, the synchronization time can be determined and included with the calculation of $T_A$.

FROMx and VOTE functions take the same amount of time to complete, since all message exchanges pass from processors to interstages and back to the processors. The effective interconnect width is 16 bits. 1 µs is required to exchange one 16 bit word, so each 240 byte FROMx and VOTE requires 120 µs. The total fault-tolerance task completion time including the loss to synchronization is:

$$T_{A_{VLSI-FTP}} = \frac{1}{17} T_0 + 600\,\mu s$$

### 8.4. FTP/FTSM Performance Analysis

Translation of the baseline design to the FTP/FTSM requires two steps. First is the addition of the FTSM and its associated fault-tolerance functions. Second, the effect of adding of a data cache is analyzed.

The determination of $T_{FTP/FTSM}$ begins by analyzing the amount of time the baseline system takes to complete the computational task. Only memory reads affect FTP/FTSM performance during the computational task, so we can divide the task time into time spent doing memory reads and time spent doing other things. For the baseline computer, this time can be expressed as:

$$T_0 = T_0 r_0 + T_0 (1 - r_0)$$

where $r_0$ is the fraction of processor time devoted to memory reads.

The performance penalty imposed by memory reads in the FTP/FTSM appears to the processor to be the result of slow memory because of the time required to exchange the symbols between channels and decode them. The processor requires a minimum number of cycles to complete a read cycle. This value is denoted $n_m$ and is equal to 3 for the 68020/68030. The FTSM system imposes a penalty of p wait states to every memory read. The total number of cycles required to complete a read access from the FTSM is $(p + n_m)$ cycles. For the FTSM system with read penalty, the task completion time is:

$$T_p = \left( T_0 r_0 \left( \frac{p+n_m}{n_m} \right) + T_0 (1-r_0) \right)$$

An independent analysis can be made of the impact of the data cache on the performance of the baseline computer. By application of the well-known cache equation [14], the task completion time for the baseline design with cache is:

$$T_{0_c} = \left( T_0 r_0 \left( \frac{q\, n_c + (1-q)\, n_m}{n_m} \right) + T_0 (1-r_0) \right)$$

where $n_c$ is the cache access time (equal to 1 for the 68030 on-chip caches) and q is the hit ratio of the cache system. Combining the last two equations gives the total time necessary to complete the computational task:

$$T'_{B_{FTSM}} = \left( T_0 r_0 \left( \frac{p+n_m}{n_m} \right) \left( \frac{q\, n_c + (1-q)\, n_m}{n_m} \right) + T_0 (1-r_0) \right)$$

This equation includes the actual computation time plus part of the fault-tolerance time. By separating the first term to yield:

$$T'_{B_{FTSM}} = \left( T_0 r_0 \left( \frac{p}{n_m} \right) \left( \frac{q\, n_c + (1-q)\, n_m}{n_m} \right) + T_0 r_0 \left( \frac{n_m}{n_m} \right) \left( \frac{q\, n_c + (1-q)\, n_m}{n_m} \right) + T_0 (1-r_0) \right)$$

the partition can be readily seen. The first term corresponds to the time spent performing the penalty cycles imposed by the FTSM. This time should be counted with the fault-tolerance task time. The remaining terms make up the computational task time.

Combining the two fault-tolerance tasks gives:

$$T_{A_{FTP/FTSM}} = T_0 r_0 \left( \frac{p}{n_m} \right) \left( \frac{q\, n_c + (1-q)\, n_m}{n_m} \right) + 163.2 \mu s$$

for the fault-tolerance task completion time for the FTP/FTSM. The 163.2 μs in the above equation represents the amount of time required to perform the four source congruency exchanges on 240 bytes of data at the beginning of the control loop. The computational task completion time is:

$$T_{B_{FTP/FTSM}} = \left( T_0 r_0 \left( \frac{q\, n_c + (1-q)\, n_m}{n_m} \right) + T_0 (1-r_0) \right)$$

The parameters in these equations must be determined before making a performance comparison. Some of the parameters are dependent on the application program and can not be determined exactly without evaluation of a real application. For these cases, estimates were made [12].

### 8.5. Performance Analysis Results

The graph in Figure 12 shows the overhead consumed by the fault-tolerance task for each FTP architecture at a given control loop iteration rate. The iteration rate is the reciprocal of the total task completion time, $T_Y$. Note that since $T_{A_Y}$ varies between the different designs, $T_{B_Y}$ is not the same on different designs for a given iteration rate. For this reason, the overhead should not be used for direct comparison between different designs. However, it does give an indication of how much processor power is available on a given design.

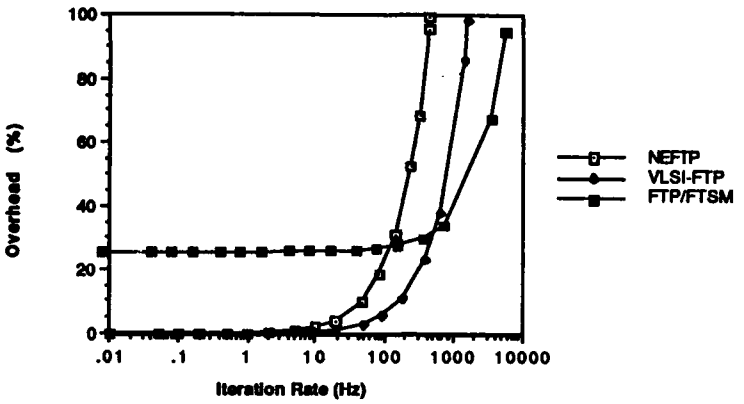



**Figure 12: Overhead vs. Iteration Rate**

The overhead in the FTP/FTSM at low frequencies is much greater than that in the NEFTP or the VLSI-FTP, because the penalty for fault masking is paid on every memory read. However, at high frequencies (500 Hz in the current design), the FTP/FTSM excels, because its constant fault-tolerance overhead, which dominates the overhead at high frequencies, is much less in the FTP/FTSM than in either the NEFTP or the VLSI-FTP. Consequently, the FTP/FTSM is more appropriate for applications requiring higher iteration rates.

Direct throughput comparisons can be made using the task completion ratios, the ratio of task completion time on a specific design to the task completion time on the baseline design. Figure 13 shows the task completion ratios for the three designs presented in this performance analysis. The x-axis is $T_0$, the task completion time on the baseline design, and the y-axis is the task completion ratio. The TCR for the baseline is always one, by definition.

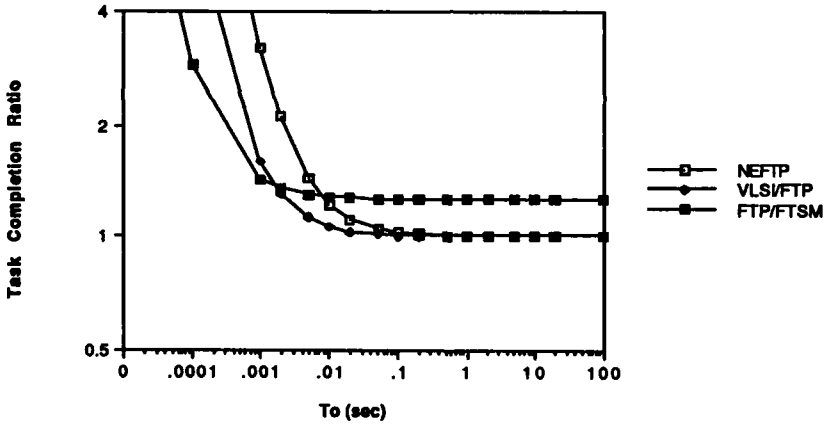



**Figure 13: Task Completion Ratios**

The TCR curve for the FTP/FTSM shows that its performance is about 80% of the baseline design for $T_0$ greater than 10 ms. The NEFTP and the VLSI-FTP both approach the raw performance of the baseline system for tasks greater than 50 ms. For tasks below 50 ms in duration, the performance of the FTP/FTSM deteriorates at a lower rate than for either of the other two designs. At 100 ms, the NEFTP and VLSI-FTP are about 25% faster than the FTP/FTSM, whereas at 1 ms, the FTP/FTSM is about 1.33 times as fast as the VLSI-FTP and about 3.30 times as fast as the NEFTP. This agrees with the observation made from the overhead graph. The FTP/FTSM is slightly worse than the other designs at low frequencies, primarily due to implicit fault masking. At high frequencies, the FTP/FTSM is much better since the constant component of the fault-tolerance task is much smaller.

The overhead in the FTP/FTSM is fairly high compared to other designs such as the NEFTP and the VLSI-FTP. The overhead limits the throughput of the FTP/FTSM at slow iteration rates to less than the NEFTP throughput. However, the NEFTP overhead begins to increase very rapidly at iteration rates above 50 Hz, whereas the overhead in the FTP/FTSM increases more slowly. As a result, we again conclude that the FTP/FTSM should have

better performance than the NEFTP or VLSI-FTP at higher iteration rates.

## 9. Conclusion

The FTP/FTSM presented in this paper represents an alternative architecture for Byzantine resilient computers. The primary benefits of the FTP/FTSM over other Byzantine resilient architectures are the elimination of memory realignment time due to the shared memory design, the improvement in short-term reliability obtained by the reduced memory requirement and the hardware implemented memory scrubber, the reduced fault latency due to the continual and implicit fault masking, and the improved high-iteration-rate performance.

The overhead due to fault-tolerance is greater than in other designs at low iteration rates. The result is that the performance of the FTP/FTSM at low iteration rates is about 80% of other similar fault-tolerant designs. For iteration rates greater than 100 Hz, the overhead in the NEFTP is greater than the overhead in the FTP/FTSM; this cross over point occurs at 500 Hz for the VLSI-FTP. Consequently, the FTP/FTSM is well-suited for applications requiring high iteration rates. The FTP/FTSM could theoretically operate at iteration rates as high as 6 kHz, although the overhead would be so high that little computation could be performed. A more reasonable limit might be 2 kHz, where the overhead is about 50%.

The reliability of the FTP/FTSM is also comparable to that of other fault-tolerant computers. The reliability of the FTP/FTSM at short mission times (around 100 hours in duration) is especially good, due to the elimination of transient errors as a significant source of potential system loss. At long mission times (on the order of 10000 hours) reliability without the possibility of reconfiguration is somewhat improved as a result of the reduction in memory as compared to a quad-FTP. However, some FTPs, the VLSI-FTP in particular, have more flexibility for reconfiguration around existing faults than does the FTP/FTSM. The quadruplicated VLSI-FTP has complete fail-op/fail-op/fail-safe capability. The FTP/FTSM can reconfigure around faults by changing the mode of the decoding operation from random to erasure, or erasure to duplex. However, the erasure mode is not guaranteed to mask all faults and the duplex mode will not mask any faults. Therefore, the current design of the FTP/FTSM is not 100% fail-safe after detection of a single fault. The current design can therefore be either fail-safe or fail-op/fail-catastrophic.

## 10. Acknowledgement

This work was supported by NASA Langley Research Center Contract No. NAS1-18565.

---


1 *Aviation Week and Space Technology*, p. 18, April 18, 1988.

2 T. Krol, "The '(4,2) Concept' Fault-Tolerant Computer," 12th International Symposium on Fault Tolerant Computing, June 1982.

3 R. Harper, *Critical Issues in Ultra-Reliable Parallel Processing*, PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA, June 1987.

4 J. Lala, L. Alger, R. Gauthier, M. Dzwonczyk, "A Fault Tolerant Processor to meet Rigorous Failure Requirements," IEEE/AIAA 7th Digital Avionics System Conference, October 1986.

5 J. Wensley, "SIFT: The Design and Analysis of a Fault Tolerant Computer for Aircraft Control," *Proc. IEEE*, 66:1240-1255, October 1978.

6 T. Abler, *A Network Element Based Fault Tolerant Processor*, MS Thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1988.

7 R. Kieckhafer, C. Walter, A. Finn, P. Thambidurai, "The MAFT Architecture for Distributed Fault Tolerance," *IEEE Trans. Computers*, 37(4):398-405, April 1988.

8 M. Pease, R. Shostak, L. Lamport, "Reaching Agreement in the Presence of Faults," *Journal of the ACM*, 27(2):228-234, April 1980.

9 D. Dolev, "The Byzantine Generals Strike Again," *Journal of Algorithms*, 3:14-30, 1982.

10 M. Fischer, N. Lynch, "A Lower Bound for the Time to Assure Interactive Consistency," *Information Processing Letters*, 14(4):183-186, June 1982.

11 D. Dolev, C. Dwork, L. Stockmeyer, *On the Minimal Synchronism Needed for Distributed Consensus*, IBM Research Report RJ 4292(46990), IBM, May 1984.

12 B. Butler, *A Fault-Tolerant Shared Memory System Architecture for a Byzantine Resilient Computer*, MS Thesis, Massachusetts Institute of Technology, p. 33-38, June 1989.

13 J. Meyer, L. Wei, "Influence of Workload on Error Recovery in Random Access Memories," *IEEE Trans. Computers*, 37(4):500-507, April 1988.

14 H. Stone, *High-Performance Computer Architecture*, Addison-Wesley, p. 31, 1987.

# A HIGHLY RELIABLE, AUTONOMOUS DATA COMMUNICATION SUBSYSTEM FOR AN ADVANCED INFORMATION PROCESSING SYSTEM

by

**Gail Nagle, Thomas Masotto and Linda Alger**
Fault-Tolerant Systems Division
The Charles Stark Draper Laboratory
Mail Stop 3E
555 Technology Square
Cambridge MA 02139
United States

## ABSTRACT

The need to meet the stringent performance and reliability requirements of advanced avionics systems has frequently led to implementations which are tailored to a specific application and are therefore difficult to modify or extend. Furthermore, many integrated flight critical systems are input/output intensive. By using a design methodology which customizes the input/output mechanism for each new application, the cost of implementing new systems becomes prohibitively expensive. One solution to this dilemma is to design computer systems and input/ output subsystems which are general purpose, but which can be easily configured to support the needs of a specific application. The Advanced Information Processing System (AIPS), currently under development at Charles Stark Draper Laboratory, has these characteristics.

This paper describes the design and implementation of the prototype I/O communication system for AIPS. AIPS addresses reliability issues related to data communications by the use of reconfigurable I/O networks. When a fault or damage event occurs, communication is restored to functioning parts of the network and the failed or damaged components are isolated. Performance issues are addressed by using a parallelized computer architecture which decouples Input/Output (I/O) redundancy management and I/O processing from the computational stream of an application. The autonomous nature of the system derives from the highly automated and independent manner in which I/O transactions are conducted for the application as well as from the fact that the hardware redundancy management is entirely transparent to the application.

## 1.0 INTRODUCTION

Reliability and performance are major concerns in the design of highly integrated flight critical systems [1]. The need to meet stringent performance and reliability requirements has frequently led to implementations which are tailored to a specific application and are therefore difficult to modify or extend. Experience with the development of modern military systems has shown that a design methodology which customizes the implementation of each new application leads to systems which are prohibitively expensive [2]. One solution to this dilemma is to design subsystems which are general purpose, but which can be easily configured to support the needs of a specific application. The performance and reliability characteristics of such a subsystem can be well established, its functionality and behavior tested and validated, and its use carefully documented. Thus it becomes an off-the-shelf component with well defined parameters which can be used as a building block of more complex systems.

### 1.1 AIPS Architecture

The Advanced Information Processing System (AIPS) is a data processing architecture comprising a set of functional building blocks which may be assembled as a fault tolerant distributed system tailored to meet the requirements of applications which need both high performance and high reliability [3]. Each building block consists of highly modular hardware and software components with well defined interfaces and validated performance and reliability characteristics. This greatly simplifies the effort required to build a data processing system for a specific application. Furthermore, the resulting system is easily maintainable, extensible, and cost effective. AIPS can serve as the core avionics system for a broad range of aerospace vehicles currently being researched and developed, including manned and unmanned space vehicles and platforms, deep space probes, commercial transports and tactical military aircraft.

The hardware building blocks which may be used in a given AIPS design are fault-tolerant, general purpose computers (GPCs), fault- and damage-tolerant inter-computer and input/output (I/O) networks, and interfaces between the networks and the general purpose computers. The software building blocks provide the services necessary in a traditional real-time computer such as task scheduling and dispatching and communication with

sensors and actuators. The software also supplies the redundancy management services necessary in a redundant computer and the services necessary in a distributed system such as inter-function communication across processing sites, management of distributed redundancy, management of networks, and migration of functions between processing sites. Figure 1 shows the laboratory engineering model for a distributed AIPS configuration currently being developed at Charles Stark Draper Laboratory [4]. This distributed AIPS configuration includes all the hardware and software building blocks mentioned earlier and was conceived to demonstrate the feasibility of the AIPS architecture.

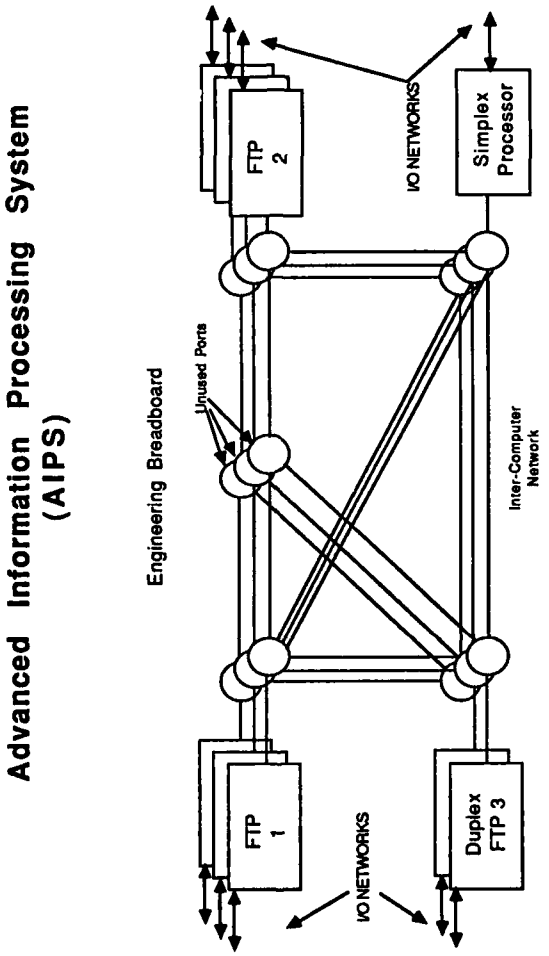



Figure 1. AIPS Distributed Configuration

### 1.2 AIPS Input/Output Networks

For communication between a GPC and I/O devices, a damage and fault tolerant network is employed [5]. Like other AIPS hardware building blocks, AIPS I/O networks are designed to provide both high throughput and high reliability. The network consists of a number of full duplex links that are interconnected by circuit switched nodes. Sensors and actuators are attached to these nodes. In steady state, the circuit switched nodes route information along a fixed communication path, or "virtual bus", without the delays which are associated with packet switched networks. Once the virtual bus is set up within the network the protocols and operation of the network are similar to typical multiplex buses. Since the hardware implementation of this "virtual bus" is a circuit-switched network, but from the GPC communication and protocol viewpoint it appears as a conventional bus, the terms "bus" and "network" are used interchangeably throughout this paper.

Although the network performs exactly as a bus, it is far more reliable and damage tolerant than a linear bus. The network architecture provides coverage for many well known failure modes which would cause a standard linear bus to either fail completely or provide service to a reduced subset of its subscribers. A single fault or limited

damage caused by weapons or electrical shorts, overheating, or localized fire can disable only a small fraction of the virtual bus, typically a node or a link connecting two nodes. The rest of the network, and the subscribers on it, can continue to operate normally. If the sensors and effectors are themselves physically dispersed for damage tolerance, and the damage event does not affect the inherent capability of the vehicle to continue to fly, then the digital system would continue to function in a normal manner or in some degraded mode as determined by sensor/effector availability.

The ability of the network to tolerate such faults comes from the design of the node. An AIPS node has five ports which can each be enabled or disabled. When the ports on either end of a link are enabled, data is routed along that link of the network. Each node in a properly configured, fault free network receives transmissions on exactly *one* of its enabled ports and simultaneously retransmits this data from all its *other* enabled ports. The nodes provide a richness of spare interconnections which can be brought into service after a hardware fault or damage event occurs.

### 1.3 AIPS Input/Output Subsystem

An AIPS computer may have access to varying numbers and types of I/O networks. The I/O networks may be global, regional or local in nature. I/O devices on the global I/O bus are available to all, or at least a majority, of the AIPS computers. Regional buses connect I/O devices in a given region to the processing sites located in their vicinity. Local buses connect a computer to the I/O devices dedicated to that computer. Figure 2 shows the topology of a 6-node regional network shared between two GPCs.

To improve performance, AIPS GPCs utilize a co-processing architecture which decouples I/O redundancy management and I/O processing from the computational stream of an application. The Computational Processor (CP) runs the application program while the I/O Processor (IOP), which is loosely synchronized with the CP, handles I/O activity. The throughput of the CP is not reduced by I/O activity, and the application is relieved of the burden of conducting I/O transactions. To an application, each I/O device appears to be memory mapped and can be referenced directly [3]. The autonomous nature of the system derives from the highly automated and independent manner in which I/O transactions are conducted for the application, as well as from the fact that the hardware redundancy management is entirely transparent to the application.

AIPS I/O System Services comprise the software modules which provide efficient and reliable communication between the user and external devices (sensors and actuators). The I/O System Service is also responsible for the fault detection, isolation and reconfiguration of the I/O network hardware and GPC/network interface hardware (I/O sequencers). I/O System Services are made up of three functional modules: the I/O Network Manager, the I/O User Interface, and I/O Communication Management. The I/O Network Manager performs hardware redundancy management for an I/O network. This software module is responsible for detecting and isolating hardware faults in I/O nodes, links, and interfaces and for reconfiguring the network to exclude any failed elements. The Network Manager function is transparent to all application users of the network. The I/O User Interface provides a user with read/write access to I/O devices or device interface units (DIUs) such that the devices appear to be memory mapped. The I/O Communication Manager provides the functions necessary to control the flow of data between a GPC and the various I/O networks used by the GPC. It performs source congruency operations on all inputs and the voting of all outputs. It also detects errors on inputs and reports communication errors to the I/O Network Manager.

These modules have been implemented in Ada for the AIPS engineering model which is currently undergoing testing, verification, and validation. Sections 2, 3, and 4 describe the functional requirements and key algorithmic features of the I/O System Service modules. Section 5 concludes with a summary of results and with suggestions for future implementations of I/O systems based on the AIPS engineering model and the experience gained in designing and building that system.

## 2.0 I/O NETWORK MANAGER

### 2.1 Network Operation and Topological Considerations

Figure 2 shows an AIPS configuration highlighting the features of an I/O network. In the figure, two GPCs are physically connected to an I/O network by means of two root links. The I/O operations on the network are conducted by the I/O Sequencer (IOS) which is controlled by the GPC through the Dual Ported Memory (DPM). The IOS is a programmable state machine which handles the low level aspects of data communication for the IOP. The network shown consists of six nodes and four DIUs. Sensors, actuators, displays and other I/O devices are attached to the DIUs.

In Figure 2 the active links, i.e. those connecting two enabled ports, are shown as solid lines. The links shown as dashed lines are spares. Transmissions are carried along enabled links only. In this topology, GPC_2 is not actively connected to the network. This is because the network shown is a local network, one whose use is dedicated to a single GPC. However, if faults were to cause a degraded mode of operation for GPC_1, the functions requiring access to the network could be migrated to GPC_2. The physical connections to GPC_2 are provided to support function migration. This topology is also capable of supporting a regional network. If this were a regional network, GPC_2 would have an active root link to the network and both GPCs would then share the resources of this network by contending for its use.

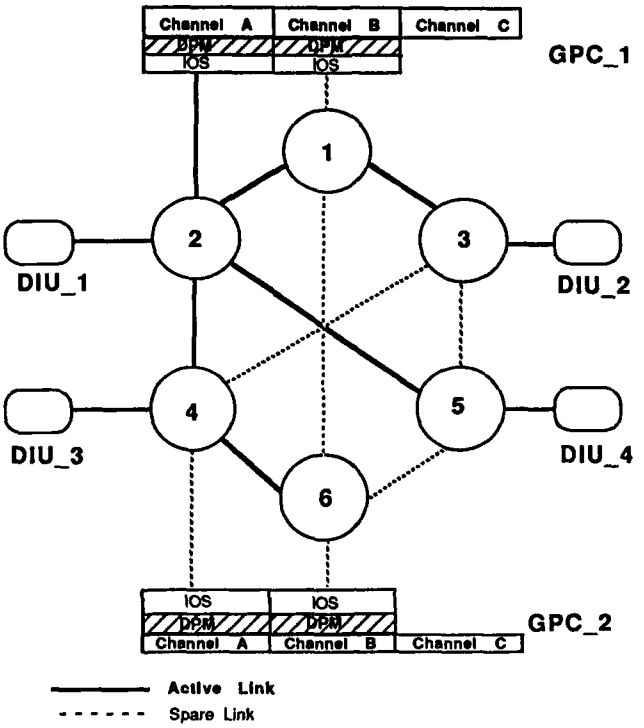



Figure 2. I/O Network With Root Links To Two GPCs

Once a properly functioning virtual bus has been established, the nodes used to form the bus remain in the active network until a component fails or is damaged. The links used to actively connect the nodes vary slowly over time which allows spare links to be brought into active service. Cycling spare links in this way provides improved coverage for latent faults.

The AIPS terminology for any communication with an I/O device is an I/O transaction. Users are provided with the ability to group I/O transactions into chains which run transactions sequentially on one network. Chains are said to be executed by the IOS because the IOS uses a simple program to effect the transmission and reception of the data associated with the transactions in a chain. Chains may be further grouped as I/O requests which cause sets of chains to run in parallel on several networks. This reduces the system overhead for obtaining correlated input/output information. It also reduces the time skew which would result from purely sequential accesses to redundant I/O devices. Users are also provided with the option of scheduling I/O requests to run periodically or on demand.

### 2.2 *I/O Network Management*

The I/O Network Manager is the software process responsible for establishing and maintaining a communication path between processors (GPCs) and Device Interface Units (DIUs) attached to the I/O network under its control [6]. Once invoked, the Network Manager has two main phases of operation: initialization and maintenance. Its activity during the initialization phase is dictated by the reason for its activation. If the Manager is invoked to manage a previously inactive network or when a graceful function migration is not possible, the Manager establishes a virtual bus within the network and performs a full set of diagnostic tests on each IOS and nodal port in the network. At the end of this initialization process, a fully tested communication path exists between all properly functioning nodes, DIUs, and GPCs in the network. This path is then capable of supporting serial communication among all functioning network subscribers. If the migration of a Network Manager from one GPC to another can be effected gracefully, data from the deactivated Manager is transferred to the newly activated Manager. Thus, if the Manager is invoked as part of a graceful function migration, the initialization phase can be reduced to a software component only. Having completed its initialization, this process notifies I/O Communication Services that the network is in service and updates the status information on this network which is available to other processes in the system.

During the maintenance phase of its operation, the Network Manager provides services on demand to the

Resource Allocator (a subprocess of the System Manager) and to the I/O Communication Manager. The Resource Allocator calls the Manager when it wishes to halt the management of this network from this GPC. This may be to effect a function migration or to support routine system maintenance. The I/O Communication Manager calls this process for one of three reasons: to repair a suspected network fault, to bring a repaired node, link or IOS back online, or to cycle spare links.

The Network Manager can detect and repair the passive failure of a node or port, the passive failure of an IOS, the failure of the channel connected to the active root link, a network component which is babbling, a node which answers to addresses other than its own, or a node which transmits on a disabled port. Once the failure mode has been determined, the network is reconfigured to remove the faulty component and restore communication to all non-failed components in the network. After the reconfiguration of the network is complete, some DIUs may be unreachable. A list of these unreachable DIUs is made available to the I/O Communication Manager when the network is put back in service. This enables it to deselect transactions to unreachable DIUs and to clear error counts against I/O devices which were temporarily out of service due to network problems.

## 2.3 Network Management Algorithms

The Network Manager contains logic which enable it to initialize or "grow" a network, to determine whether or not the network is functioning properly, and to reconfigure the network to remove a failed component. An overview of some of the algorithms employed in this logic is presented below.

### 2.3.1 Network Growth

Network growth is the process whereby the links between the nodes in the network are enabled to form a virtual bus which supports communication among network subscribers (GPCs and DIUs). Data flow in the network is controlled by the configuration of the ports in each node. When a node receives a message addressed to itself on any port, disabled or enabled, it carries out the command encoded in the message and then transmits its status from all its enabled ports, including the port which received the message if that port is enabled.

Nodes are added one by one to the virtual bus. The algorithm used to add these nodes causes the bus to expand in a treelike manner. For proper operation, there can be no loops in the active network. The growth algorithm generates a maximally branching, minimum length path to every node in the network. This configuration is later changed in order to cycle spare links and to repair faults. The detection of a protocol violation when any new link is called into service results in the disabling of that link. Furthermore, the growth algorithm employs a set of diagnostic tests which exercise every link in the network, including spare links. The tests can also detect the presence of some malicious failure modes such as nodes which transmit on disabled ports and nodes which respond to commands addressed to other nodes. Hence, the growth algorithm produces a very robust communication path. In addition to joining network nodes into a virtual bus, the growth process is also concerned with enabling communication paths to network subscribers: DIUs and remote GPCs. This is accomplished by enabling nodal ports adjacent to these devices and determining whether or not these components obey the protocols established for all functioning network components. The detection of protocol violations results in the connection to the subscriber being disabled.

Network growth begins by establishing an active root link to one of the root nodes (a node adjacent to an IOS) and ensuring that this root node has a port which can be used as the springboard to the rest of the nodes in the network. If an active root link is found, the remaining nodes are added to the active tree. Any node which is not connected to the active tree after this stage is complete is unreachable. After the nodal network is established through the active root link, the spare root links to the network must be enabled and tested. In order to establish spare root links, the inboard port of each active root node is enabled. In a similar manner links are activated to connect the DIUs and remote GPCs of a regional network to the network. Finally, status is collected from all nodes in the network to verify that no additional failures have occurred in the network during the growth process. If no discrepancies are found, the node status chain is updated by removing transactions to nodes which have been identified as failed.

### 2.3.2 Fault Analysis

If the I/O Communication Manager detects errors after executing a chain for an application program, it invokes the Network Manager. The Network Manager executes a chain to collect status reports from each node. It then analyzes status information provided by the IOS and the status reports collected from the nodes to identify the type of fault and the network element suspected of producing the error. Three types of analysis are performed: raw data analysis, error analysis, and node data analysis.

While carrying out its principal function of sending and receiving data, the IOS detects various error conditions on the network. The IOS imparts this information to the processor through several status registers and through a buffer of status information appended to the incoming data of every transaction. This information is referred to as raw data. By analyzing this information certain failure modes are identified. If failed components are present, the class of fault is reported. This can be a failed root link, a failed link or node, or a babbler. Whenever errors are detected, the root link is switched to be sure that the error is not attributable to a failed IOS. When the raw data analysis is completed, the results of the analysis and the status reports from non-failed nodes are passed on to error and data analysis for further processing.

Error analysis is the process of deducing which network element produced the errors. If all the nodes in the

network have errors, error analysis attributes the errors to a root link failure. If one or more nodes have errors, two possible failure modes are considered: a single node failure or a failed link or node. The single node failure symptom could be indicative of a node which does not respond to commands but which continues to retransmit messages. It could also be a node which itself is not failed but to whose address another node in the network responds. The single node failure is easy to diagnose since exactly one node in the status collection chain shows an error. If more than one node has errors, the remaining problem is to determine if the observed errors fit the pattern for a failed link or node. The signature of such a failure is when all nodes which have errors form a treelike pattern downline of the failed link or node. If the error symptoms do not indicate one of these failure modes, the fault is undiagnosable.

Data analysis is the process whereby the status information returned by the nodes is examined. In particular, this analysis identifies a node which is transmitting from a port which should be disabled. Specifically, if a non-failed disabled port reports the reception of a valid message, the node adjacent to that port is transmitting from a disabled port. (Adjacent ports are always in the same configuration, either both enabled or both disabled.)

### 2.3.3 Reconfiguration

The purpose of reconfiguration is to restore error-free communication to all reachable, non-failed nodes in the network. The reconfiguration action depends on the type of failure as determined by fault analysis. The result of fault analysis is actually a hypothesis about what is causing the errors on the network. The reconfiguration process tests this hypothesis by reconfiguring the network to isolate the component suspected of producing the error symptoms and then verifies that the network is again fully operational. Therefore, the network may go through several intermediate configurations before the reconfiguration process is complete.

Network fault analysis identifies six classes of faults: a root link failure, a babbler, a link or node failure, a node which transmits from a disabled port, a single node failure, and an undiagnosable failure. A separate strategy exists to deal with each of these fault classes. The reconfiguration process is considered complete when the node status chain is executed on the reconfigured network and does not detect any errors.

When the fault hypothesis is a failed root link, a spare root link is chosen to establish a new connection to the network. The new interface is then used to execute the node status collection chain. If no errors are detected, as would be expected in the case of a passive failure involving only the IOS, the reconfiguration process is complete. If either a babbler or a link failure is detected, the reconfiguration process starts over with a new root link dealing with a new fault. This behavior would be expected for an active fault such as a babbling IOS or a passively failed root node which now must be removed from the network so that service to nodes downline of it can be restored.

When a babbler or a stuck on high condition is detected detected by the IOS at its receiving interface to the network, the network is regrown using the fast grow option. This growth procedure does not perform the extensive diagnostic tests that the initial growth algorithm requires.

A failed node generates the same error pattern as a failed link. Thus, when the fault analysis reveals the presence of this failure mode, the reconfiguration algorithm must determine which fault has actually occurred and reconfigure the network accordingly. It is first assumed that a link has failed. The failed link is disconnected and an attempt to reach the failed node, i.e. the node immediately downline from the link, is made by using any spare ports on that node which are adjacent to nodes not in the failed node list. When this strategy fails to restore communication with the failed node (possibly because no spare ports are available), each branch of the failed tree must be reconnected to the active network. Only one successful connection to any spare port on a branch needs to be made in order to restore communication to the entire branch (and possibly to the failed node and all other nodes in the failed tree). A three transaction chain is used to reconnect the branch to the network. The first two transactions enable the ports on either side of the new link while the third transaction disables the former inboard port of the failed node in case the node adjacent to that inboard port is a babbler. If the failed node correctly returns its status, the repair is complete and the absence of errors is verified by collecting status from every node in the network. If the failed node is still not reachable, the port connecting this node to the present branch is disconnected and the proper functioning of the newly enabled link is verified. The net effect of this process is to restore communication with all reachable nodes in the network while isolating the failed node. As communication to each branch is restored, the possible pool of spare links increases. Thus if any branch was not connected because of a lack of spare links, this branch is retried whenever a connection to another branch is successful. Any nodes which are still unreachable at the end of this process are marked failed.

If a node retransmits valid data from a port which should be disabled, the entire node must be isolated from the network. This failure mode is distinguished from a babbler which is always transmitting a random bit stream or is stuck on high. When a babbling port is identified, the adjacent port of the neighboring node is disabled. This neighboring node will not retransmit from its other enabled ports anything received by the disabled port. Furthermore, the node will ignore any random bit patterns it receives. However, if the neighboring node receives a request for status addressed to itself on a disabled port, it will transmit its status from all its enabled ports, even though it does not retransmit the initial request. If the failed node is not removed, each time the manager asks for status from the node adjacent to this port, it would receive two valid commands to report its status. The failed node is isolated from the network by disabling the ports on nodes immediately adjacent to it. Isolating a node is a simple matter if the node is a leaf; only the link connecting it to the network needs to be disconnected. Otherwise, the nodes downline from the failed node need to be reconnected to the network through alternate links. If the node to

be isolated is the current root node, a new root link is selected. The link connecting the inboard port of the failed node to the network is then disabled. Next, an attempt is made to reestablish a connection to each isolated branch via a spare link to a node in that branch. Only one such connection needs to be made to restore communication to all the nodes in the branch. After the new connection is enabled, the link connecting the failed node to this branch is disconnected. This algorithm, while isolating the failed node, restores communication to every reachable node in the network. Nodes which cannot be reached because earlier failures have depleted the pool of spare links are marked failed.

Figure 3 illustrates the steps needed to isolate a node from the network. Suppose that Node 2 is to be removed from the network. First the link connecting Node 2 to Node 1 is disabled (A). When this step is completed, Nodes 2, 3, 4, 5, and 6 are also isolated from the GPC as shown in part II. Node 2 is the root of a tree with two branches, each of which must be reconnected in turn. By enabling the link between Nodes 1 and 6 (B) and disconnecting the link between Nodes 2 and 4 (C), one of these branches is reconnected to the active network as shown in part III. Finally, a link is enabled between Nodes 5 and 6 (D) and the link between Nodes 2 and 3 (E) is disabled. In this reconfiguration, Node 2 is isolated while preserving several links in the network. In larger networks, the performance gain of this approach over regrowth of the entire network is significant.A single node failure can occur if the failed node is a leaf node, if its retransmission function still works correctly but its status reporting capability is impaired, or if another node is responding to this node's address, making it appear that this node is failed. If the failed node is the current root node, before proceeding, a new root link is selected from the available spares, status is collected using this new root link, and a new error analysis is performed. The failed node is then isolated from the network, as described above however, care is taken not to address this node directly because of the possibility of an addressing problem. When the node is isolated, this node is again queried for its status. If a valid response is received, indicating the presence of a node which responds to the addresses of other nodes, the network is regrown with a full set of diagnostic tests to isolate this faulty node. Otherwise, an attempt is made to find an alternate route to this node using any port except its previously failed inboard port.

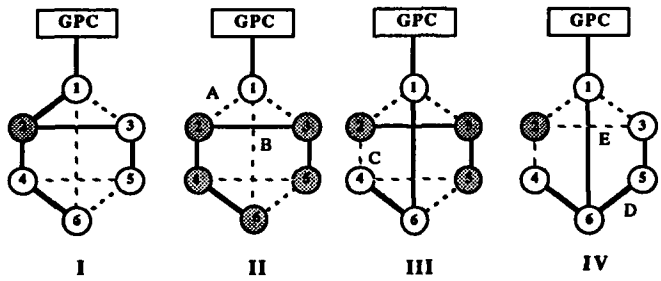



Figure 3. Removing A Node And Reconnecting Its Branches

The stratagem for dealing with undiagnosable errors is network regrowth. Regrowth is also the back up reconfiguration strategy used when two attempts to reconfigure the network have not succeeded in eliminating errors.

## 3.0 I/O User Interface

To satisfy the performance requirements of modern avionics systems, AIPS uses a co-processing architecture to provide input/output services.[7] A goal of the design and implementation of AIPS I/O System Services is to make the co-processing architecture transparent to the user. I/O devices, which are attached to fault tolerant networks, are accessed by application programs by means of system calls which simulate memory mapped I/O. Figure 4 shows a functional view of the AIPS co-processing scheme. The I/O User Interface is a process which is resident on the CP; its companion process, I/O Communication Services (discussed in Section 4), is resident on the IOP. The I/O User Interface accepts system calls from an application program and communicates via shared memory with the I/O Communication Services to provide the requested service. These services include read/write access to I/O devices and scheduling and synchronization of I/O activity.

The I/O User Interface provides a flexible communication framework which can be used by a system designer to tailor I/O activity for a particular application. The basic unit of communication on an I/O network is a *transaction*. A transaction is a command transmission to a device on a network optionally followed by a response from that device. An *output transaction* does not require a response; an *input transaction* does. An application can communicate with a single device on an I/O network or it can group transactions into a *chain* to allow transactions to be executed sequentially on a single network as a unit. Furthermore, it provides the means to form *I/O requests* from chains. An I/O request is a set of one or more I/O chains, each of which executes simultaneously on a different I/O network. These I/O request specifications result in CP/IOP shared memory assignments for data and

error information for each transaction. In addition, the I/O User Interface provides system calls for scheduling I/O requests and safely accessing the shared memory locations. Although I/O devices may be connected to multiple fault tolerant networks, all network access protocols, source congruency and error processing on inputs, and fault masking on outputs are transparent to the user.

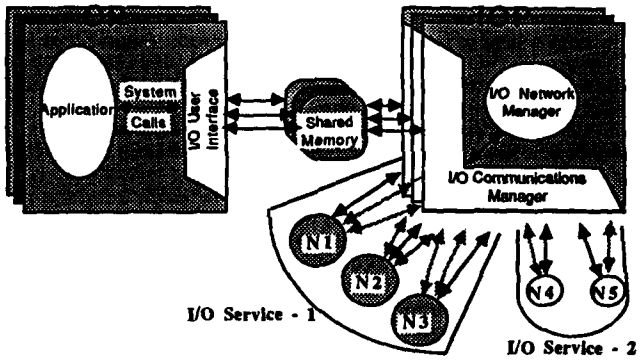



Figure 4. AIPS I/O System Services resident on a triplex GPC with two I/O Services. The figure highlights functional aspects of the AIPS co-processing architecture.

The I/O User Interface is divided into three functions: I/O Request Construction, I/O Data Access Operations, and I/O Request Scheduling. The I/O Request Construction function allows the user to create I/O transactions, specify how they will be grouped and how each I/O request will be scheduled. The I/O Data Access Operations provide the read/write routines that allow the user to access I/O chain data in shared memory, while hiding the CP/IOP protocol from the user. The I/O Request Scheduling provides the user with the flexibility to schedule each I/O request as a cyclic free running task that runs and signals the caller when each cycle has completed or an on-demand task that is only scheduled when requested.

### 3.1 I/O Request Construction

Using a knowledge of the distribution of sensors and actuators among the various networks in a given AIPS implementation, the application designer determines how best to organize the I/O transactions required by the application. Critical design issues include overall system performance, synchronization between the application and its I/O requests, and sensor/actuator redundancy. In order to set up the I/O framework for an application, the applications designer specifies I/O transactions first, followed by chains and finally I/O requests.

Because AIPS is intended to serve a wide range of avionics applications, the I/O User Interface provides the application designer a very flexible input/output environment. I/O Request Construction affords the user a wide range of options in specifying I/O transactions. For example, a user can specify that a transaction be either input or output since output transactions do not require a response. The number of transactions and the length their corresponding outgoing and incoming messages is only constrained by the physical size of the DPM. When specifying the system, the system designer can select the size for this memory. The outgoing data for a transaction may be designated as dynamic or static data. This feature is provided in order to decrease chain execution time. For example, a command to obtain a sensor reading may be static while an actuator command would typically be dynamic. Static data is copied into the IOS only once whereas dynamic data is copied prior to each execution of the associated chain. In the case of an input transaction, the user must also specify the transaction time-out which is the maximum time that can elapse before the first byte of data is received from the DIU. This parameter is provided by the user since different devices may take different amounts of time to respond to a command. If the time-out expires, a fault in the network is indicated. The fault may be due to a failed network component or to a failed DIU. If the I/O Network Manager can reconfigure the network to repair the fault, the time-out condition will not recur. However, if the fault is due to a failed DIU, every I/O request which references that DIU will trigger error processing. To reduce this overhead in the event of a DIU failure, the user must specify the maximum number of errors that are tolerable before the transaction is bypassed, i.e. removed from the chain by the I/O System Services.

A user also has several options when specifying I/O requests. The user must specify an I/O request time-out which is analogous to the transaction time-out described above. Furthermore, the I/O User Interface provides a range of scheduling options as well. An I/O request may be executed periodically or on demand. Users may prioritize their I/O requests to allow requests with a higher priority to take precedence on the network whenever two or more requests are scheduled at the same time. In addition, if the I/O request is periodic, the user can specify the repetition period, when to start the I/O request, and whether or not to stop it. The user can also specify whether an

active or a passive signal is to be used to indicate the completion of an I/O request. Additional details concerning scheduling options are given in Section 3.3.

### 3.2 I/O Data Access Operations

The I/O Data Access Operations provide routines that allow the user easy access to I/O devices. The redundancy management of the fault-tolerant network, network access protocols, source congruency and error processing on inputs, fault masking on outputs, and the CP/IOP communications protocol are transparent to the user. The applications user is able to write commands to the DIUs on a transaction by transaction, chain by chain, or request by request basis. Data returned by the DIUs can be read at similar levels of granularity. Other I/O Data Access routines provide the user with error information at the transaction, chain and I/O request level. This capability expedites error processing and allows the user to discard data which may contain communication errors. If an I/O request cannot be executed when the scheduling requirements have been fulfilled, the application and the I/O request become desynchronized. In this case an overrun error indicator is provided. In addition, other system calls allow the user to add (i.e. select) and remove (i.e. deselect) transactions from their corresponding chains. For example, users can deselect transactions from devices known to be faulty, thereby facilitating error recovery. Similarly, a user can select transactions to support dynamic I/O request reconfiguration.

The actual mechanics used to support the I/O Data Access routines require the I/O System Service routines on the IOP and CP to communicate through shared memory by means of a strict protocol. Semaphores and double buffering are used to maintain consistent data sets.

### 3.3 I/O Request Scheduling

The I/O Request Scheduling provides the application with the flexibility to schedule each I/O request as a cyclic free running task that runs and signals the caller when complete or as an on-demand task that is only scheduled when requested. The user specifies the scheduling requirements for the I/O requests when the I/O requests are created. On demand I/O requests are started on the IOP only when the user issues a start command. The periodic I/O requests may be scheduled to start on demand, at a specific time, or after a specific amount of time has expired. The period for each request is specified by the user and is not constrained in any way. Moreover, periodic I/O requests may be scheduled to run forever or to stop on demand.

I/O Request Scheduling provides two synchronization mechanisms to coordinate the I/O requests with the application tasks: events and flags. Events are active signals which are observed by the GPC Real Time Operating System. The events interrupt the CP and result in the activation or deactivation of an application task. Flags are passive signals which may be observed or ignored by the application tasks. The flags do not interrupt the co-processor and are used to indicate the completion of the I/O requests. Whenever an I/O request completes, the I/O System Service on the IOP sets a flag in shared memory. The application tasks on the CP use system calls to read and clear the flag. The completion of an I/O request is indicated by an event only if the user has specified this option when creating the I/O request. The event is used to activate an application task which is blocked pending the completion of its I/O request.

## 4.0 I/O COMMUNICATIONS MANAGEMENT

I/O Communications Management provides the functionality needed to control the flow of data between a GPC and the various I/O networks it uses [7]. These functions are divided into two categories, I/O Traffic Control and I/O Low Level Utilities. I/O Traffic Control executes I/O requests. This involves selecting the I/O request which will run on a given network, transferring data between shared memory and the dual ported memory of the IOS, and processing any errors detected during the execution of the I/O request. In addition, I/O Traffic Control coordinates the simultaneous execution of chains for I/O requests which use several networks. I/O Low Level Utilities are responsible for distributing congruent inputs to the redundant channels of the GPC, voting output data to provide fault masking, and screening input data for errors. This error processing involves both error detection and error logging.

### 4.1 I/O Traffic Control

I/O Traffic Control initializes I/O System Services on the IOP and coordinates all subsequent I/O activity, especially the scheduling and selection of I/O requests for each network on a GPC. This process handles both the execution of the chains of the I/O requests and the error processing and logging required by the I/O requests.

I/O requests are conducted on an I/O service. An I/O service is a logical organization imposed on the physical networks to which a GPC is connected. An I/O service may be provided by a regional network, i.e. one which is shared among several GPCs, or a local network. When an I/O service is local, it may involve a set of networks. I/O requests are sorted into prioritized queues for execution on an I/O service; the priority of the request is specified by the user. Each I/O service has a corresponding Queue Manager whose primary role is to control access to its I/O service. In addition, some requests for network usage are generated by the system for the purpose of system maintenance, such as requests for spare link cycling and network component restoration. These requests are serviced by the Queue Manager when no I/O requests are pending.

A scheduling mechanism drives the execution of I/O requests. Based on the scheduling parameters specified by the user, a "posting task" is created for each I/O request. Whenever the scheduling requirements of a given I/O request are met, the associated posting task posts its I/O request to the Queue Manager of the appropriate I/O service. The system may also post requests for I/O service. As the requests arrive, they are queued by priority

pending network availability. In certain cases the arrival of an I/O request may cause the pre-emption of the request currently executing on the network. For example, an I/O request for an application will pre-empt a request for spare link cycling. Furthermore, I/O requests which have the *highest* priority level in the system will also pre-empt I/O requests of lower priority.

Whenever an I/O service is idle, the Queue Manager selects the pending I/O request with the highest priority. Prior to execution of the I/O request, the Queue Manager handles any transaction selection and deselection requests. When processing an I/O request for an application, the Queue Manager transfers dynamic output data from the shared memory to the IOS memory, executes the chains comprising the request on the appropriate networks, and then processes the responses resulting from any input transactions in the chain. If a fault in an I/O network causes errors to occur, the Queue Manager takes the appropriate network out of service and calls the Network Manager to perform network fault detection, identification, and reconfiguration.

### 4.2 I/O Low Level Utilities

The I/O Low Level Utilities handle a variety of hardware intensive operations for I/O System Services. These include the use of the AIPS data exchange hardware which ensures source congruency on inputs to the GPC and fault masking on outputs sent to I/O devices. Another set of hardware intensive functions are required to control the operation of the IOS and to process error information returned by the IOS following chain execution. I/O Low Level Utilities also provide operations which allow other I/O System Services to obtain information about network topologies as specified by the system designer. This information is stored in logical form in an I/O Database maintained by I/O Low Level Utilities.

## 5.0 CONCLUSIONS

The Advanced Information Processing System (AIPS) I/O System Services have been designed, implemented, and tested on the centralized configuration of the AIPS engineering model. The I/O Network Manager manipulates the large number of possible interconnections between the circuit switched nodes to maximize the system's overall reliability and survivability. The responsibilities of this software include the following: initializing a network; performing network FDIR when required; periodically cycling spare links to reduce fault latency; and re-establishing connections to nodes which have been repaired. The I/O User Interface provides a flexible and user friendly environment in which a system designer can readily tailor the I/O activity for a specific application. It also affords an application programmer the simplicity of virtual memory mapped access to I/O devices. The I/O Communications Management system interacts with the I/O User Interface to schedule, execute, and process the I/O requests. The cooperation of these processes makes such underlying complexities as the system's redundancy, distributed nature, variable complement of resources and fault recovery transparent to the user.

Initial testing of the network management software was done by randomly injecting faults into the links, nodes, root nodes and IOSs. The network status display and error logs were used to monitor these tests. The software correctly identified and reconfigured the network in all tests. The preliminary testing of the I/O User Interface and the I/O Communication Manager was performed by creating a sample set of I/O requests and application tasks. The I/O requests exercised all of the I/O communication, scheduling and synchronization features of the I/O User Interface. The application tasks tested the interprocessor communication of I/O data and status information. The testing of I/O System Services focused on the near simultaneous execution of I/O chains on redundant networks in the presence of faults on one or more of the networks. In all test cases, the faults were identified, the network was reconfigured, and the I/O requests were executed on schedule.

Unlike other AIPS building blocks, I/O networks are not Byzantine Resilient. Therefore, they are not demonstrably resilient to malicious faults, so the network validation process does not benefit from the theoretical rigor of the Byzantine Resilience approach to fault tolerance. Thus, future designs and implementations for network FDIR will include the addition of communication protocols which use authentication techniques to verify both the contents and sender of a message. Such protocols reduce the requirements which a system must meet in order to achieve Byzantine Resilience to malicious faults. [8]

## 6.0 ACKNOWLEDGEMENT

This work was supported by NASA Langley Research Center under contracts NAS1-17666 and NAS1-18565.

## 7.0 REFERENCES

1. G.C. Cohen, C.W. Lee, L.D. Brock, and J.G. Allen, "Design and Validation Concept for the Integrated Airframe/Propulsion Control System Architecture," NASA Contractor Report 178084, June, 1986.

2. J.J. Deyst and L.D. Brock, "Modular Avionics Systems Studies," A Collection of Technical Papers from the AIAA/IEEE 8th Digital Avionics Systems Conference, San Jose, California, October, 1988, pp. 1-7.

3. J.H. Lala, "Advanced Information Processing System," Proceedings of the AIAA/IEEE 6th Digital Avionics Systems Conference, Baltimore, Maryland, December, 1984, pp. 199-210.

4. G. Nagle, L. Alger, and A. Kemp, "Advanced Information Processing System: Input/Output Network Management Software," NASA Contractor Report 181678, May, 1988.

5. J.H. Lala, A. Ray, R. Harper, "A Fault and Damage Tolerant Network for an Advanced Transport Aircraft," American Automatic Control Conference, San Diego, CA, June, 1984.

6. G. Nagle, "An Ada Implementation of the Network Manager for the Advanced Information Processing System," The First International Conference on Ada Programming Language Applications for the NASA Space Station, Houston, TX, June, 1986.

7. T. Masotto, L. Alger, "Advanced Information Processing System: Input/Output System Services," NASA Contractor Report 181874, August, 1989.

8. L. Lamport, R. Shostak, M. Pease, "The Byzantine Generals Problem," ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, pp. 120-126.

# Formalizing Developments: from Theory to Practice.

*Karim BECHANE and Michel LEMOINE*
ONERA-CERT
Département d'Etudes et de Recherches en Informatique
31055 Toulouse CEDEX France
2, avenue E. Belin
Phone:+33-61-55-70-73
UUCP: bechane@tls-cs.cert.fr , lemoine@tls-cs.cert.fr

**Abstract:**

***The need to support the whole process of system development with available and really efficient environments is a major challenge of the software technology. In this paper we briefly recall the old paradigms in use. Then we introduce a new one which is well supported by an environment under consideration. In this environment we emphasize the use of the DEVA language which aims at:***

- ***expressing formal developments.***
- ***expressing and using development methods.***
- ***finally re-using formal developments for the derivation of new developments and consequently new programs.***

***A realistic example based on the JSP method is introduced, then formalized and expressed in terms of DEVA. The correctness of the run guarantees that the resulting programs satisfies their initial specifications.***

## 1 Introduction

Computing is a discipline which is now widespread and crucial in a multitude of domains of our daily activities, especially in security, dangerous and meticulous operations. Certain number of computer utilizations require a 100% reliable and correct software. Consequently computer scientists have established a real discipline of programming known as a Science of Programming[Gri81].

## 2 The Software Production

Every time we have to produce a program dealing with a lot of parameters and functions it is not obvious for the human–being to keep in mind at the same time all these parameters. So he/she easily looses the control of the development with no premeditation. This gets worse when more than one designer or programmer are involved in the task (because the problem of communication between them). The problem is sometimes due to the misunderstanding of the requirements. The client perhaps expressed badly his/her problem or used a confused or inconsistent language (natural language for instance). An available solution to this problem is through the use of a formal or a semi–formal (graphical) language, allowing "good" (precise, safe, consistent, ...) specifications.

The conversion from informal requirements to programs is done manually according to the SLC (Software Life Cycle) Waterfall model as presented in figure1[Bal81]. It leads eventually to errors and programs which do not satisfy the initial requirements. One of the main drawbacks of the preceding paradigm is in the maintenance process. Indeed, any error detected too late can conduct to a total re-design.

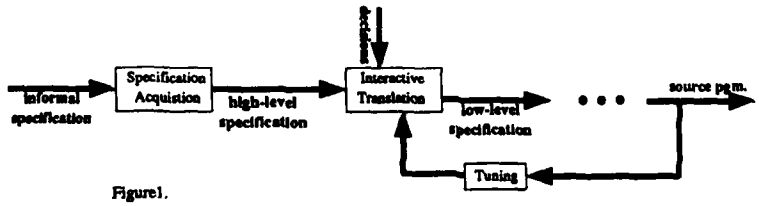



Figure1.

A more elaborated paradigm (see Figure2) [Bal81] for developing software consists of designing automatic systems integrated in the process of software production, where interactive facilities are given to bridge the gap between a high-level specification and a low-level specification. This leads to the following paradigm, in which formal development is emphasized by using formal representations and transformations:

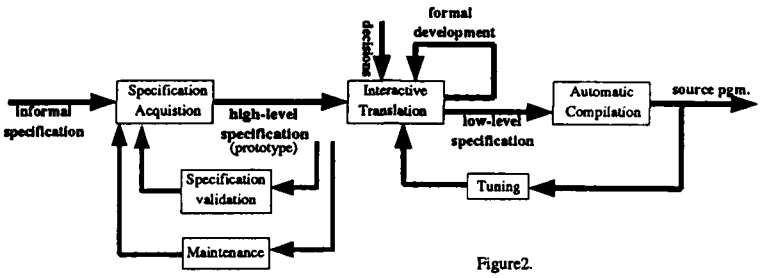



Figure2.

On the contrary of the Waterfall model, this SLC model proposes:

- to validate the high-level specification
- to assist the low-level specification development.

Once the initial formal specification written and adopted as being the mirror of the problem, there are two manners to materialize the corresponding program(s).

**The transformational techniques**

The initial specification is modified and transformed into another equivalent specification from the functional standpoint. This last one has in plus the property to be closer to the implementation. The process of transformation is repeated until reaching the totally implementable specification *ie.* a program.

This technique can be achieved by :

- an assisted mode
  This first mode used by Burstall and Darlington [BD77] consists of rewriting automatically a certain scheme (skeleton) of specification into another scheme. The drawback of this technique is the limited number of such rewriting schemes. Its advantage is the possibility to computerize it. In fact this operation of transformation is not totally automatic since we have to provide some heuristics to control the unification between the problem and the available schemes.
- a manual mode
  In this second mode, the designer gives himself the second specification supposed to be equivalent to the first. The advantage of this approach is the non-limited number of refining specifications (representations) that can be proposed. The drawback is working out proof obligations after every new proposed specification to check equivalence between the two representations. This is done for example in the Vienna Development Method [Bjo89] or in the Abstract Machines approach[Abr88].

**The verification technique**

Having the first specification of the problem a program is directly written. The verification of the correspondence between the program and the specification can be done either by submitting the program to a series of tests (this solution is expensive and not convincing at all), or by using formal verification techniques such as Floyd-Hoare [Hoa69] of the programs. Nevertheless, their use is limited practically to the stage of programs.

## 3 Software Development Method/Process

A method consists of rules for organizing and guiding an activity: the software development process for our concern. It must provide instructions about what to do next. Through a certain number of steps a method, more precisely a formal method, will allow to derive a correct program from the specification. Thus a method insures a good result *ie.* a program satisfying its specification.

An ambitious activity would be the treatment of the whole development process as a formal object. In this spirit a formal description of existing software development methods would be a first step toward rigorous implementation of tools or environments and so extent toward the automation of those methods.

A method specifies in an explicit and detailed manner possibly non deterministic sequence of activities. These activities manipulate objects.

Current real-world development methods are rarely known to be formal or formalizable. the method-supports are too limited. All the general knowledge and human reasoning (*eg.* heuristics, ...) are not explicitly described.

It is shown that the easiest part in the formalization of a software method concerns the temporal structure of the method, *eg.* the order in which activities are to be performed. This can be formally described within a hierarchical framework. On the other hand objects used in methods seem formalizable at least *wrt.* to some of their properties and constraints [DG88].

## 3.1 Reusability

The inability of the software industry to quickly produce software systems of high quality has diminished the ability to take advantage of the current increase in hardware productions. The main focus of software engineering research has been on increasing software quality. There is also the need for increasing software productivity. One technique is software reusability. Under the general term of "reusability" are five important subtopics[Jon84]:

- reusable data
- reusable architecture
- reusable design
- reusable programs and common systems
- reusable modules

Most of the current approaches are based on empirical methods such as keywords or description in natural language. For some specific fields there exist good libraries of software components. Their descriptions are given in the terminology of the application area. However there is no general approach to this problem. Some papers suggests some criterions for software reusability which are theoretically well founded. The formal specification appear to be more suitable as basis for the retrieval of reusable software components than informal specifications. The criterions used are not completely constructive, but they provide a guideline to find out reusable software components and prove their reuse. [GM88] treats the reusability of code through algebraic and hierarchical specifications. The approach described in [Luq87] depends on normalizing specifications to reduce the variations in the representation of software concepts.

## 3.2 Formal Expression of Developments

During the last years, computer scientists have been interested in finding a means to express what their programs will do, what will be their properties and how indeed produce good ones. To realize the preceding objectives a mathematical framework is necessary. This led to invent formal languages and mathematical based constructs and notations to write the specifications. This permits the specifications to be consistent, complete from the initial specification, provable and transformable *ie.* possible to operate on them formal transformations.

# 4 The Tooluse Project

The general objectives of the ToolUse project[1][CJL+89] are to provide active assistance in the various activities of software development through the formalization, and the support, of development methods. This formalization is done through a language, DEVA, used to express the design decisions related to methods as well as a specification language.

## DEVA : requirements and design

The requirements for the design of DEVA were that it should formally express the derivation (handling, transformation, adaptation, assembling) from specifications to programs. A more precise and technical requirement is the ability for DEVA to express design decisions by constructions as well as by intentions. This has led to a language which is a partial synthesis of $\lambda$-calculus, natural deduction and constructive logic systems. A hopeful consequence of this technical direction is that the same framework is able to express specifications, programs and developments.
The chosen approach has as theoretical basis the work done around Nederpelt's $\Lambda$- language [Ned73], as well as more recent work on the Calculus of Constructions [Coq87] and on Veritas[HD86].
Without entering into deep technical descriptions, we could say that DEVA is a high–order typed $\lambda$-calculus. Thus DEVA allows one to express the formation and inference rules of object theories (it is used as a kind of meta-language).

### DEVA: definition

In DEVA there are two classes of objects:

- Texts, which describe development expressions. A text may be assigned a type, expressing the result of an achieved development. The recursive definition is limited by the pre-existence of an un-typable text constant: **primal**.
- Contexts, which describe theories on which the developments are based. The contexts are used to introduce modularity.

---

[1] ToolUse is partially supported by the ESPRIT Programme.

The basic components of contexts are:

- text declaration: **x : t**
- text definition: **x := t**
- context definition: **part x := ct**
- context importation: **import ct**

The main formation rules for texts are:

- a primitive constant: **primal**
- context abstraction over a text: **[ct ⊢ t ]**
- application of a text: **t1(t2)**
- judgement: **t1 cert t2** to assert that t1 is of type t2

The next version of DEVA will include additional operators to express control objects as texts:

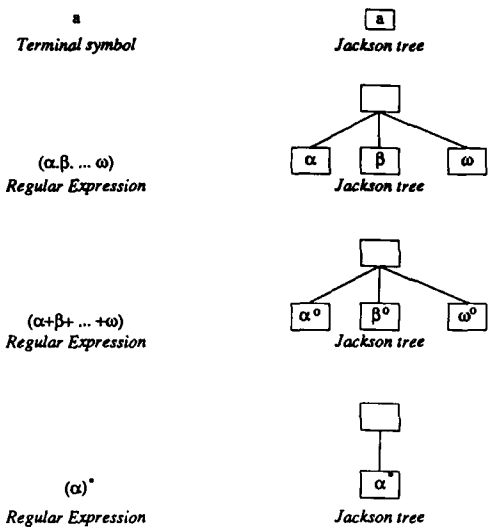
- sequential composition
- nondeterministic choice
- an iteration operator
- a matching facility

## 5 A Brief Presentation of JSP method: the case study

In the present section we'll use the concepts presented above through a particular example of a JSP development. Our intention is not to propose a new formalization of the Jackson's method, but the use of an existing one presented in [Viv86][Ngu88] The JSP method of program design gives a systematic way of solving a wide class of data processing problems. It identifies a number of types of problem, in an intuitive way, and describes a method of solution for each. The basic method requires that the following three steps are carried out:

1. the inputs and outputs are described precisely, by means of tree structured diagrams
2. the correspondences are identified between nodes on the input and output trees
3. a program is constructed in which each statement is associated with a particular input node and its corresponding output.

The Jackson's trees are represented by regular languages[Hug79]. A single terminal symbol is represented by a tree consisting of a node, labeled by the symbol. An expression which is a concatenation of symbols or bracketed sub-expressions is represented by a node with an ordered sequence of sons in which the $i$-th son is the root of a tree representing the $i$-th concatenated symbol or bracketed expression. An expression which is a union of symbols or bracketed sub-expressions is represented by a node with an ordered set of sons in which each son is the root of a tree representing a symbol or bracketed sub-expression in the union, and there is a son representing each alternative. Each alternative is marked with the symbol °. An expression which is an iteration of a symbol or bracketed sub-expression is represented by a node with one son and labeled * which is the root of the tree for iterated symbol or bracketed sub-expression.

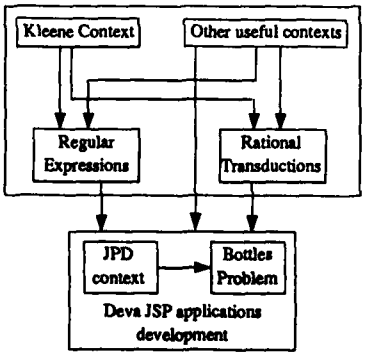



The JSP method requires that correspondences are identified by inspection between a pair of tree representing the input and output. In terms of regular expressions this implies correspondences between symbols or sub-expressions in the input and output regular expressions. The correspondence defines a desired translation of nodes on the input tree into nodes of output tree.

After creating data structures and after they are combined by pattern-matching of components called correspondences formalized as rational transductions [Dyb87], the detailed specification of the program is extracted (conversion of the specification into a text form). It consists of the allocation of physical program functions called elementary operations and for conditions necessary for the procedural logic, in accordance with the specific features of the target language.

*Note:*In the following , **regplus, regpoint, regstar** stand respectively for + , . , * for regular expressions operators in an infix form.
**ratplus, ratpoint, ratstar**stand for respectively for + , . , * for rational transductions operators in an infix form.

The overall structure of the example is given by the following figure:



```
%-------------------------------------------------------------%
%   Definition of the Rational Transductions Context          %
%-------------------------------------------------------------%
    part RatTrans := (|
              RatTr      :  primal
              & import part RegExp
              & import rename PI    to ratPI,
                       PHI   to ratPHI,
                       star  to ratstar,
                       point to ratpoint,
                       plus  to ratplus
               in part Kleene(RatTr)

          & (.) => (.) : [ RegE & RegE |- RatTr ]


          & RULE1  : [ R1,R2,S1,S2:RegE |-
[ regpoint(R1,R2) => regpoint(S1,S2) |- ratpoint((R1 => S1),(R2 => S2)) ]]

          & RULE2  : [ R1,R2,S1,S2:RegE |-
[ regplus(R1,R2) => regplus(S1,S2) |- ratplus(R1 => S1,R2 => S2) ]]

          & RULE3  : [ R,S:RegE         |-
  [ regstar(R) => regstar(S) |-  ratstar(R => S) ]]

                |)
```

The JSP method consists of a series of atomic developments between two specifications of the whole process of the development. The definition of the specification due to[Viv86] is 6-uplet $\Sigma$ = **(RE,RS,TA,TI,TL,RT)** where

**RE** is the input system of rational transductions
**RS** is the output system of rational transductions
**TA** is the set of authorized atomic rational transductions
**TI** is a set of of forbidden atomic rational transductions
**TL** is the set of non-refined rational transductions
**RE** is a set of equations of rational transductions.

```
%-------------------------------------------------------%
%         Definition of the JSP context                 %
%-------------------------------------------------------%
      part JSP := (|
        spec         : primal
      & nat          : primal
      & import part RegExp
      & import part RatTrans


      & TSeqs        : primal
      & TemptySeq    : TSeqs
      & Taddel       : [ RatTr & TSeqs  |- TSeqs]
      & Trmvel       : [ nat   & TSeqs  |- TSeqs]
      & Ttakel       : [ nat   & TSeqs  |- RatTr]

                  ......................

      & CRtakel : [ nat   & CRSeqs          |- CpleR]

  & mkspec  : [ CRSeqs & CRSeqs & TSeqs & TSeqs & TSeqs & CTSeqs |- spec]

                |)
```

Let us introduce the example of the bottles presented in[Viv86]. The informal specification of the problem is as follows:

*Each bottle come on a traveling band to be filled and is verified.*
*A bottle comes empty and then is filled.*
*The controller must decide if the bottle is full or not.*
*The empty bottle is weight and then full.*
*The net weight must be computed to decide if a bottle is full enough.*

```
part bouteilles := (|
       import part JSP
       & bouteilles,
       bouteille,
       choix,
       vide,
       remplissage,
       pleine,
       acceptee,
       refusee,
       poids,
       mesures,
       tare,
       poidstotal,
       calcul,
       poidsnet        :  RegE
```

where $RE$ and $RS$ are respectively:

$RE$ :

```
        %-----------------------------------------------
        |          bouteilles = ( bouteille.choix)*     |
        |          bouteille  = vide.remplissage.pleine |
        |          choix      = acceptee + refusee      |
        -----------------------------------------------%
  RE := CRaddel( bouteilles, regstar(regpoint(bouteille,choix)),
 ( CRaddel( bouteille, regpoint(vide,regpoint(remplissage,vide)),
   ( CRaddel( choix,regplus(acceptee,refusee),CRemptySeq)))))
```

$RS$ :

```
        %-----------------------------------------------
        |          poids    = ( mesures.calcul)*        |
        |          mesures  = tare.poidstotal           |
        |          calcul   = poidsnet                  |
        -----------------------------------------------%
 RS  := CRaddel ( poids, regstar(regpoint(mesures,calcul)),
          ( CRaddel( mesures, regpoint(tare,poidstotal),
            ( CRaddel( calcul, poidsnet, CRemptySeq)))))
```

The different formal atomic specifications of the bottles-problem are :

$\Sigma_i = (RE, RS, TA, TI, TL_i, RT_i)$ where

$TA = \{(empty \rightarrow tare), (filling \rightarrow \lambda), (full \rightarrow total), (accepted \rightarrow netweight), (refused \rightarrow netweight)\}$

$TI = \emptyset$

$TL_0 = \{bottles \rightarrow weight\}$

$RT_0 = \emptyset$

which are translated in Deva into :

```
    &  TII      := TemptySeq
    &  TAA      := Taddel( vide => tare,
                     ( Taddel( remplissage => regPI,
                     ( Taddel( pleine => poidstotal,
                     ( Taddel( acceptee =>  poidsnet,
                     ( Taddel( refusee => poidsnet, TemptySeq)))))))))
```

```
&  TL0      := Taddel( bouteilles => poids ,TemptySeq)
&  RT0      := CTemptySeq
```

Using the Jackson's rules (see RatTrans context), the first specification $\Sigma_0$ is transformed into $\Sigma_1$ and then $\Sigma_1$ into $\Sigma_2$ *etc.* until the last $\Sigma_4$ in which $TL_4$ contains only those elements of $TA$, with the condition that it does not contain elements of $TI$ (this is not the case because $TI = \emptyset$).

By performing rewritings on $TL$'s element(s) the different $TL_i$ and $RT_i$ (the only elements modified in the different transformed specifications) are successively:

$TL_1 = \{bottle.output \rightarrow measures.calculus\}$

$RT_1 = \{(bottles \rightarrow weights, bottle.output \rightarrow measures.calculus)\}$

```
& TL1 := Taddel ( (regpoint(bouteille,choix)=> regpoint(mesures,calcul)),
                   TemptySeq)
& T1 : regstar(regpoint(bouteille,choix)) => regstar(regpoint(mesures,calcul))
& ( RULE3(regpoint(bouteille,choix),regpoint(mesures,calcul)T1)
                cert
    ratstar( regpoint(bouteille,choix)=>regpoint(mesures,calcul)))
& RT1 := CTaddel( ( bouteilles => poids),
      ratstar( (regpoint(bouteille,choix) =>regpoint(mesures,calcul))),
                 CTemptySeq)
```

$TL_2 = \{(bottle \rightarrow measures; output \rightarrow calculus)\}$

$RT_2 = RT_1 + \{((bottle.output \rightarrow measures.calculus), (bottle \rightarrow measures; output \rightarrow calculus)\}$

$TL_3 = \{(empty \rightarrow tare); (filling \rightarrow \lambda); (full \rightarrow total); (output \rightarrow calculus)\}$

$RT_3 = RT_2 + \{((bottle \rightarrow measures), (empty \rightarrow tare); (filling \rightarrow \lambda); (full \rightarrow total))\}$

and

$TL_4 = \{(empty \rightarrow tare); (feeling \rightarrow \lambda); (full \rightarrow total); (accepted \rightarrow netweight); (refused \rightarrow netweight)\}$

$RT_4 = \{((output \rightarrow calculus), (accepted \rightarrow netweight + refused \rightarrow netweight))\}$

```
&  T4 : ((regplus(acceptee,refusee)) => regplus(poidsnet,poidsnet))
&  R41  := (RULE2( acceptee,refusee,poidsnet,poidsnet,T4))
&  R42  := ( ratplus((acceptee => poidsnet),(refusee => poidsnet)))
&  ( R41 cert R42)

&  TL4 := Taddel( (vide => tare),
                 ( Taddel( (remplissage => regPI),
                 ( Taddel( (pleine => poidstotal),
                 ( Taddel( (acceptee => poidsnet),
                 ( Taddel( (refusee => poidsnet), TemptySeq))))))))

&  RT4  := CTaddel( (choix => calcul),
     ( ratplus( (acceptee => poidsnet),(refusee => poidsnet))),RT3)
```

The four atomic specifications are then given by:

```
&  spec0 := mkspec(REE,RSS,TAA,TII,TL0,RT0)
&  spec1 := mkspec(REE,RSS,TAA,TII,TL1,RT1)
&  spec1 := mkspec(REE,RSS,TAA,TII,TL1,RT1)
&  spec3 := mkspec(REE,RSS,TAA,TII,TL3,RT3)
&  spec4 := mkspec(REE,RSS,TAA,TII,TL4,RT4)
```

These atomic specifications constitute successively the complete development of the Bottle Problem through the JSP development method.

## 6 Conclusion

DEVA is a higher-order language which allows to express different sort of objects such as development, steps of developments, and programs which are results of developments.

Through the case study of expressing part of the JSP method in the DEVA framework, the objective was twofold:

1. to show how DEVA language is able to express developments and their proofs.

2. to show how formal techniques can capture the methodological knowledge of a development.

Even if it is at a mumbling stage, the formal discipline of software development have shown that the issue is promising and will be in the near future an industrial practice and usage.

## Bibliography


[Abr88] J.R. Abrial. Abstract Machines. Technical report, —, 1988.

[Bal81] R. Balzer. Transformational Implementation:An Eemple. *IEEE Transactions on Software Engineering SE-7*, 7(1):3–14, 1981.

[BD77] R.M. Burstall and J. Darlington. A Transformation System for Developing Recursive Programs. *JACM*, 24(1):44–67, 1977.

[Bjo89] D. Bjorner. Towards a Meaning of 'M' in VDM: Specification Methodology Aspects of the Vienna Development Method. In LNCS, editor, *TAPSOFT'89*, 1989.

[CJL+89] J. Cazin, R. Jacquart, M. Lemoine, P. Michel, and P. Maurice. Method Driven Programming. *IFIP–Congress San Francisco*, pages 351–356, August 1989.

[Coq87] Th. Coquand. *Une Théorie des Constructions*. PhD thesis, Université Paris VII, Jan. 1987.

[DG88] D. Dzierzgowski and E. Grégoire. Formalizing Software Development Methods. In IEEE, editor, *System Design: Concepts Methods and Tools*, 1988.

[Dyb87] P. Dybjer. From Type Theoryn to LCF – a Case Study in Program Verification. In P. Dybjer et al., editor, *Proc. of the Workshop on Programming Logic*, October 1987.

[GM88] M.C. Gaudel and T. Moineau. A Theory of Software Reusability. In LNCS, editor, *ESOP'88 Nancy France*, volume 300, March 1988.

[Gri81] D. Gries. -. In *The Science of Programming*. Springer-Verlag, Berlin-New-York, 1981.

[HD86] F. Hanna and N. Daeche. Purely functional implementation of a logic. In $8^{th}$ *Int. Conf. on Automated Deduction*. SPRINGER–VERLAG, LNCS230, 1986.

[Hoa69] C.A.R. Hoare. An Axiomatic Basis for Computer Programming. *CACM*, 12(10):576–583, 1969.

[Hug79] J.W. Hughes. A Formalization and Explication of the Michael Jackson Method of Program Design. *Software–Practice and Experience*, 9:191–202, 1979.

[Jon84] T.C. Jones. Reusability in Programming: A Survey of the State of the Art. *IEEE Transactions on Software Engineering*, 10(5):488–497, 1984.

[Luq87] Luqi. Normalized Specifications for Identifying Reusable Software. In *Exploring Technology: Today and Tomorrow – Dallas Texas*, 1987.

[Ned73] R. P. Nederpelt. *Strong Normalization for a Typed Lambda Calculus with Lambda Structured Types*. PhD thesis, Eindhoven, 1973.

[Ngu88] T.T. Nguyen. On Jackson's Structured Programming Method. Technical report, UCL ToolUse.TD.TN88/1a, Feb. 1988.

[Viv86] F. Vivarès. Prolégomènes à un langage de développement pour la méthode de Jackson, JSD. Technical report, ONERA–CERT, November 1986.

# METHODOLOGIE DE DECOMPOSITION D'APPLICATION DE NAVIGATION CRITIQUE EN ELEMENTS SIMPLES

par

**Bernard Chavana** (Chef de projet Logiciel) Tel: 75 79 85 11 et
**François de Sainte Maresville** (Responsable de Groupe Assurance Qualité) Tel: 75 79 87 28
SEXTANT AVIONIQUE/CROUZET
22, Rue Jules Védrines
26027 Valence
France

RESUME

Le but du document est d'exposer la démarche de spécification/conception utilisée dans le cas du développement d'un produit critique pour la sécurité d'un aéronef.

L'exposé n'aborde les aspects réalisation que dans le sens spécification des moyens de réalisation (codage, test...)

Il montre comment à chaque étape, on s'est attaché à trouver une démarche simple pour assumer les contraintes du projet.

PLAN



## 1 - CONTEXTE DE DEVELOPPEMENT

### 1-1 - PRESENTATION DU PROJET

Le système de référence primaire SRP destiné à l'Hélicoptère SUPER PUMA MK2 est un système de base ayant pour but de fournir à l'instrumentation de bord (Visualisation, Pilote Automatique, Navigation), Les informations dites de références primaires c'est à dire :

- Le cap magnétique,
- Les attitudes,
- Les vitesses angulaires en axe porteur,
- Forces spécifiques,
- Vitesse air,
- Altitude-Pression,
- Vitesse verticale anémobarométrique,
- Température extérieure.

Il y a deux systèmes SRP identiques redondants.

Un système SRP est constitué principalement :
- . D'un FDC (Flight data computer)
- . D'un HSU (Heading sensor unit)
- . D'un PSU (Pressure sensor unit)
- . D'un TPU (Temperature unit).

La fonction principale du FDC est du type AHRS (Attitude and Heading reference system).

1-2 - CONTRAINTES

La fonctionnalité SRP est considérée comme critique pour la sécurité de l'appareil.
Le SUPER PUMA MK2 étant soumis à un programme de certification, les bases de certification revendiquées sont :

. FAR 29 Amendement 16
. Condition spéciale DGAC "Protection contre le foudroiement"
. Critères de navigabilité IFR suivant la lettre FAA du 15/12/78.
. Normes DO 160 B et MIL 810 D, 461 B, 462 pour les conditions d'environnement.

Le suivi du processus de développement est effectué par le CEAT.

1-3 - HYPOTHESES DE DEPART

Cette fonctionnalité critique (c'est à dire probabilité d'occurence de défaut inférieure à 10-9 pour une heure de vol) a été ramenée au développement de deux chaines essentielles (c'est à dire probabilité d'occurence de défaut inférieure à 10-6 pour une heure de vol).

Voir Annexe 1.

En ce qui concerne le logiciel, le guide méthodologique étant la DO 178 A, deux possibilités se présentaient :

. Deux équipes séparées développant chacune un logiciel essentiel,
. Une équipe développant un logiciel critique.

Pour des raisons de coût de développement et de mise en oeuvre :

. Duplication de toutes les ressources,
. Problèmes de détection des modes communs
. Délais tendus (16 mois).

On s'est orienté vers le développement d'un logiciel symétrique de niveau critique selon le DO 178 A.

2 - LES CHOIX DANS LA DEMARCHE DE DEVELOPPEMENT DU LOGICIEL

Voir Annexe 1.

L'exposé porte uniquement sur la partie descendante du cycle de vie.

On remarque cependant que l'aspect réalisation (partie montante) est largement dépendant des options prises initialement.

En particulier, l'aspect vérification validation et test en général qui constitue la majeure partie des phases de réalisation, a été un facteur déterminant pendant les phases de spécifications.

La décomposition du logiciel en éléments simples a pour but d'obtenir un logiciel testable et maintenable.

Cette démarche est suivie et adaptée à toutes les phases de la définition du produit (spécification, conception) et permet aussi pour chaque étape, de définir les moyens spécifiques adéquats.

2-1 - PHASE DE SPECIFICATION

2-1-a - SPECIFICATION TECHNIQUE PRODUIT

Voir Annexe 1.
L'élaboration de la spécification technique débute par l'analyse de la spécification technique de besoin qui a pour origine le service MARKETING et/ou le CLIENT.

La spécification de besoin constitue la définition externe du produit :
On y trouve en particulier les informations suivantes :

- principales fonctionnalités
- masse encombrement
- conditions d'environnement
- prix objectif.

On en déduit l'architecture matérielle par un processus itératif dans lequel entre en considération l'ensemble des contraintes suivantes :

- Coût objectif
- Performances
- Fiabilité, sécurité, maintenabilité.

L'activité se traduit par :

. La définition des caractéristiques et des performances au niveau du système et de chacun de ses sous-ensembles.
. L'orientation des choix :

- Au niveau système : Redondance des fonctions

- Au niveau de chaque sous-ensemble :
  . Logiciel critique,
  . Unicité des types de processeurs,
  . Unicité des moyens de production et logiciels de base.

. Orientation des choix temps réel :
Chaque processeur est considéré comme une fonction de transfert (bloc fonctionnel homogène) linéaire vis à vis de flots d'informations d'entrée.

L'architecture choisie repose sur des Processeurs Parallèles synchronisés et sur une Communication inter-carte par un bus parallèle.

- La mise en place du groupe projet :

L'ensemble des intervenants majeurs du projet constitue le groupe projet.
Le choix des intervenants est directement lié aux impératifs du projet et fait partie des moyens à mettre en place pour assurer le bon développement du produit.

2-1-b - SPECIFICATION DU LOGICIEL

Cette phase a pour but :

- La répartition des exigences des spécifications techniques entre les différentes cartes.
- La décomposition en fonctions élémentaires sur les différents processeurs et identification des communications.

CRITERES :

- Minimiser le couplage inter-fonction
- Baser la décomposition sur l'analyse de la transformation des flots de données
- Faciliter la testabilité par :
  . La spécification d'exigences vérifiables :
    - Précision de calcul définie au niveau de chaque flot de données,
    - Enchainement des modes de fonctionnement suivant les commandes opérateur et les états du système clairement définis sous forme d'automate avec contraintes de temps pour chaque transition.
  . La traçabilité et la cohérence de ces exigences : justification par des matrices de conformité.
  . L'introduction de points de visibilité dans le programme : Définition de flots de vérification qui n'ont pas d'utilité opérationnelle mais qui permettent de faciliter la validation et la maintenance des entités fonctionnelles.

CRITERES D'ARRET :

- Toutes les exigences fonctionnelles doivent être couvertes ou identifiées et bornées.
- Tous les critères choisis pour la phase doivent être respecté..

METHODE DE VERIFICATION :

- Analyse des documents et établissement de matrices de conformité et références croisées.
- Revue de fin de phase avec participation du client et du CEAT.

CARACTERISTIQUE DE L'ACTIVITE :

L'activité de spécification logiciel se traduit par la vision statique du problème à résoudre.

Les aspects dynamiques concernent le fonctionnel (gestion des modes, délais maximum entre entrée et sortie etc).

Cette activité constitue le cadre des exigences pour la phase de conception.
Exemple :

Répartition des processeurs et décomposition de la fonction AHRS
Voir Annexe 2.

## 2-2 - PHASE DE CONCEPTION DU LOGICIEL

Cette phase a été décomposée en deux activités :

- Conception préliminaire ou globale
- Conception détaillée.

Cette décomposition n'apparaît pas dans la DO 178 A mais on la retrouve dans le DOD 2167.

L'intérêt de cette décomposition est de consolider très tôt l'architecture du logiciel et de bien séparer les niveaux de détails dans la documentation.

Ceci a pour conséquence de rendre l'accès aux informations progressif et clair.

### 2-2-a - CONCEPTION PRELIMINAIRE

La conception préliminaire ou globale a pour but :

- De prendre en compte et de répartir les exigences des spécifications logiciel
- De définir l'architecture dynamique du logiciel
- D'effectuer la décomposition des fonctions et sous fonctions identifiées lors de la phase de spécification en modules élémentaires : chaque module et son interface sont alors définis.

CRITERES :

- Minimiser le couplage inter-module
- Respecter la communication inter-fonction.

Si celle-ci doit être modifiée au niveau de la conception, elle doit l'être également au niveau de la spécification pour maintenir la traçabilité entre les documents.

- Traçabilité avec la spécification logiciel
- Ségrégation des types de traitement et de la communication :
- Maîtrise du temps réel
- Répartition Moniteur/Application.

Le moniteur se charge :
- . De l'ensemble de la gestion des interruptions,
- . Du pilotage des coupleurs externes,
- . De la communication inter-tâches.
- . De la gestion des tâches.

L'ensemble de la complexité du temps réel est ramené au niveau du moniteur.
Celui-ci doit être aisément testable.
Il a donc été conçu pour minimiser les chemins de test.

Les options suivantes ont donc été prises :

- . C'est un SEQUENCEUR
- . Il traite des TACHES CYCLIQUES.

L'application ne traite que du fonctionnel :
Un langage supporté par un outil spécifique permet d'interfacer l'application avec le moniteur en permettant la description des caractéristiques de l'application :

Description des tâches :
- . Nom, type, fréquence, activation, priorité...
- . Communications : inter-tâches (flots de données)
- . Description des coupleurs d'E/S : Références du Handler, adresse physique du coupleur, N° d'IT.

- L'architecture doit être issue d'une décomposition hiérarchique descendante des fonctions et sous-fonctions.

  Elle est constituée de fonctions, sous fonctions, séquences (tâches cycliques) et modules.

  Le processus doit être déterministe :
  L'enchainement des taches, les préemptions doivent être définis.
  Les seuls évènements pouvant modifier le temps réel de façon aléatoire sont les interruptions originaires des coupleurs d'entrée/sortie ou les exceptions levées par l'application en cas de problème.

. Assurer la testabilité par :

  Des exigences vérifiables : précision de calcul définie au niveau de chaque flot, temps d'exécution défini pour chaque séquence.

Chaque composant logiciel est constitué de son code et de sa documentation.
La documentation est constituée par l'entête du module.
Au moment de la conception préliminaire, le module possède une entête partiellement renseignée.

CRITERES D'ARRET :

- Toutes les exigences de la spécification logiciel doivent être prises en compte.
- Les critères respectés, les règles de conception détaillée sont affinées pour la phase suivante (document SDS software design standard).

METHODE DE VERIFICATION :

Analyse des documents et établissement de matrices de conformité et références croisées.
Revue avec participation du client.

CARACTERISTIQUE DE L'ACTIVITE :

L'activité de conception préliminaire se traduit par la vision dynamique du programme.

2-2-b - CONCEPTION DETAILLEE

La conception détaillée a pour but :

- De prendre en compte et de répartir les exigences de la conception préliminaire
- De décrire finement les traitements que doivent effectuer chaque composant de l'arborescence issue de la conception préliminaire. Le composant étant une unité de compilation.

CRITERES :

- La présentation des composants est régie par un ensemble de règles.
  Ces règles sont établies au plus tard pendant la conception préliminaire dans le document SDS. Elles concernent :

  . les rubriques de l'entête des modules
  . Le pseudo langage (structure)
  . Les contraintes spécifiques (traitement des cas d'exception, temps d'exécution.
  . Les objectifs de l'activité de test.

Le composant logiciel est constitué de son code et de sa documentation.
La documentation est constituée par l'entête du module.
Au moment de la conception détaillée chaque module possède une entête complètement remplie.

- Les cas de test doivent être spécifiés au niveau de chaque composant dans un fichier spécifique.
- Traçabilité avec la conception préliminaire
  . pas de module supplémentaire.
  . pas d'exigence supplémentaire.

CRITERES D'ARRET :

- Toutes les exigences de la conception préliminaire sont prises en compte au niveau de chaque composant.
- Les critères de la phase sont respectés.

METHODE DE VERIFICATION :

Analyse des documents et établissement de matrices de conformité et références croisées.
Revue avec la participation du client et du CEAT.

CARACTERISTIQUE DE L'ACTIVITE :

L'activité de conception détaillée se traduit par la mise en place de l'ensemble des composants logiciels et de leurs spécifications (pseudo code).

## 3 - METHODE ET MOYENS SPECIFIQUES

### 3-1 - CHOIX PRELIMINAIRES

Le développement d'un logiciel critique nécessite une maitrise totale :
- du cycle de vie en terme de processus de génération (moyens, méthodes et assurance qualité) depuis la spécification jusqu'à l'exécutable.
- de la configuration.

Le choix des outils constitue un problème délicat qui repose sur des critères de sécurité et de perénnité.

En bref chaque outil doit avoir un niveau de confiance suffisant. La DO 178 A fait référence au crédit que l'on peut apporter aux outils.

Le choix du langage et des chaines de développement associées repose sur la maîtrise du code généré autant que sur les performances induites par l'utilisation de celui-ci.

Les outils développés dans le cadre de l'affaire ont été conçus de telle sorte qu'ils n'influent pas sur le code sujet au processus de certification.

Ainsi les chaines de test automatiques génèrent elles des fichiers résultats consultables laissant la sanction sur la conformité à une personne.

Le crédit apporté à ces chaines repose donc sur la manière dont elles ont été testées initialement mais aussi sur leur comportement pendant le développement.

L'ensemble des outils spécifiques développés autour du projet ont fait l'objet de spécificationset sont également gérés en configuration.

Tout outil spécifique doit être simple.

Deux solutions se présentaient pour la définition de l'atelier logiciel SRP :

- Des outils séparés,
- Une structure d'accueil ayant la possibilité d'intégrer tous les outils.

C'est cette option qui a été choisie par l'utilisation de la structure d'accueil PALAS.

LA STRUCTURE D'ACCUEIL PALAS :

L'environnement PALAS permet de réaliser et de gérer la configuration de tous les composants du programme.
PALAS accueille des chaines de productions spécifiques prenant en compte toutes les contraintes d'assurance qualité adaptées :

- Maîtrise de la configuration et des évolutions
- Maîtrise de la structure du logiciel.

Toute modification d'un composant le rend périmé ainsi que tous les composants dont il dépend d'où :

- Maîtrise de l'activité de test,
- Maîtrise du processus de production.

Tous les outils sont appelés par des commandes normalisées.

A côté des services de bases fournis par PALAS, les choix suivants ont été faits afin d'améliorer la maîtrise du développement :

- Plusieurs classes de composants illustrant leur type de fonctionnalité et leur niveau dans la hiérarchie.
- Toutes les classes doivent être testables.
- L'ensemble des cas de tests associés a tout composant est géré en configuration et doit être rejouable lors de chaque modification du composant.
- Contrôle du processus de production sur la base de l'intégration BOTTOM-UP incrémentale.
- Maîtrise des interfaces.

Le contrôle sur les données est très fort : toute modification d'un flot rend anormale la configuration liée au producteur et à tous les utilisateurs de ce flot.

STRUCTURE PALAS ET CHAINES DE DEVELOPPEMENT :

Le besoin :

- Plusieurs classes de composants illustrant leur type de fonctionnalité et leur niveau dans la hiérarchie.

Ce besoin s'est traduit par la définition des classes suivantes :

Classe APPLICATION :
Cette classe correspond au sommet de l'arborescence et comporte le fichier exécutable opérationnel sur la cible.

Classe MINIAPPLICATION :
Cette classe correspond également au sommet d'une arborescence et comporte un exécutable recouvrant partiellement les fonctionnalités de l'application.
Les fonctions non implantées sont remplacées par des composants "bouchons".

Classe FONCTION :
Cette classe correspond a une fonctionnalité identifiée au niveau de la spécification du logiciel.
Elle est constituée exclusivement d'un exécutable de test.

Classe SOUS FONCTION :
Cette classe correspond à la décomposition des fonctionnalités de la spécification du logiciel.
Elle est composée d'un environnement de test pour cible simulée :

Classe SEQUENCE :
Cette classe correspond à une tache cyclique.

Classe MODULE :
Cette classe correspond à l'ensemble des composants situés sous la hiérarchie de la séquence.
Elle est composée :
- . du code source
- . du code objet
- . du lanceur des tests
- . du descriptif des cas de tests
- . du fichier résultat des tests
- . du verrou logiciel.

Le besoin :

- Toutes les classes sont testables :
  L'ensemble des cas de tests associés a tout composant est géré en configuration et doit être rejouable lors de chaque modification du composant.
  Toutes les classes comportent des objets de test.
  Toute modification du code source entraine la destruction du fichier résultat de test et détruit l'objet TSTOK constituant le verrou logiciel.

Le besoin :

- Contrôle du processus de production sur la base de l'intégration BOTTOM-UP incrémentale.

  La construction de l'application nécessite la présence de tous les objets prévus au moment de la définition de la structure PALAS.

Le besoin :

- Maîtrise des interfaces :

  Le contrôle sur les données est très fort : toute modification d'un flot rend périmés le producteur et tous les utilisateurs de ce flot.

  La description des flots de données est incluse dans l'interface de l'entité gérée par PALAS.

  Toute modification d'interface rend impossible l'utilisation de tous les composants ayant une dépendance sur cet interface.

3-2 - ACTIVITE DE VERIFICATION ET VALIDATION

Les modalités de déroulement de cette activité sont spécifiées avec les objectifs suivants et en complète harmonie avec la démarche de décomposition :

- Maîtrise de l'activité de test en gérant simultanément l'objet et son environnement de test.

  Chaines de production spécifiques assurant la cohérence des tests avec les objets à tester et permettant de tester chaque module au niveau fonctionnel et structurel.

- Processus d'intégration incrémental intégré dans l'environnement de production ainsi toute configuration tentant d'intégrer un composant non testé est impossible.

L'ensemble de ces possibilités reposent sur l'analyse de l'application en conception préliminaire.

4 - CONCLUSION

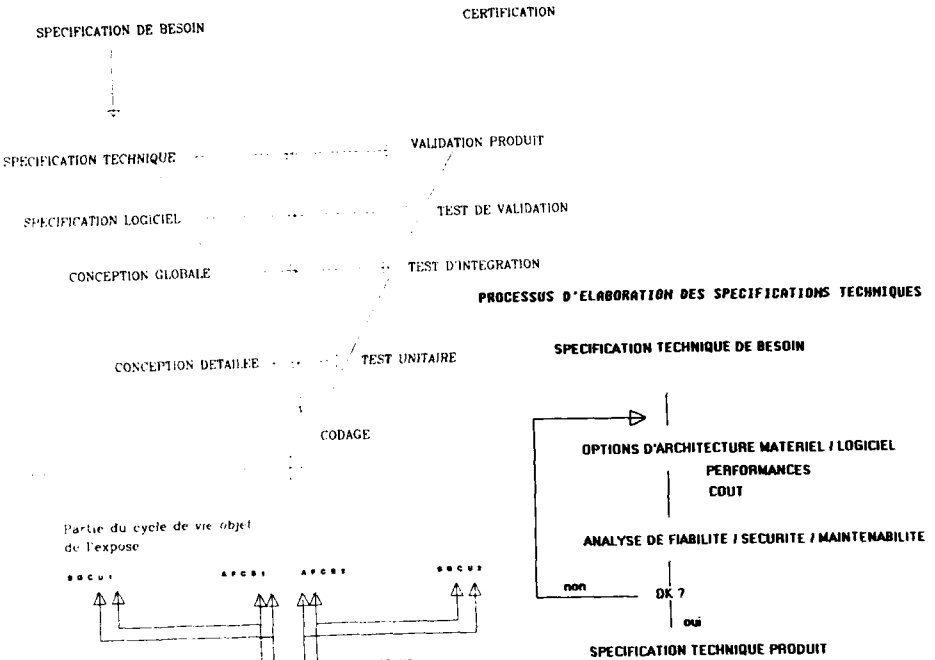
La ligne de conduite du projet a largement bénéficié de la sensibilisation de toute l'équipe aux contraintes induites par la criticité du produit.

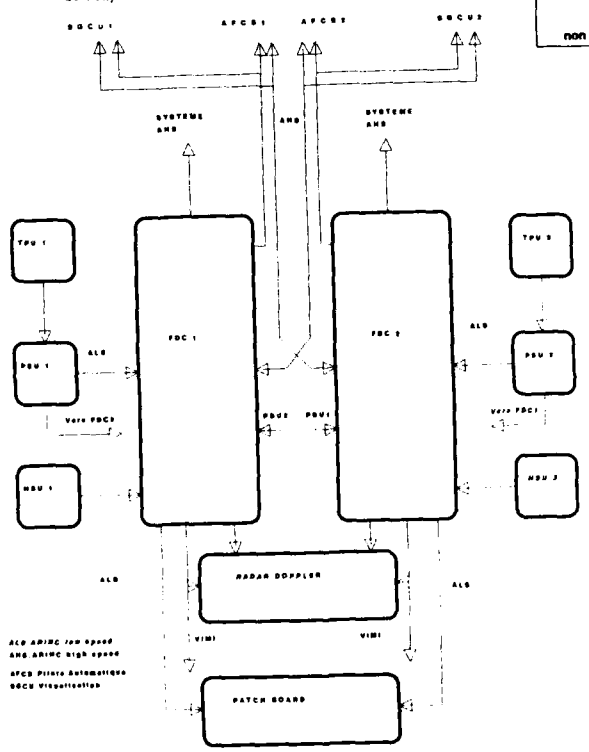
Ainsi à partir du moment ou le haut niveau d'assurance qualité était défini par des règles strictes, fondées sur la démonstration et la justification, la méthodologie a été naturellement appliquée.

Son application nous a confirmé l'importance des tâches amont et en particulier la définition de l'environnement de production.

ANNEXE 1

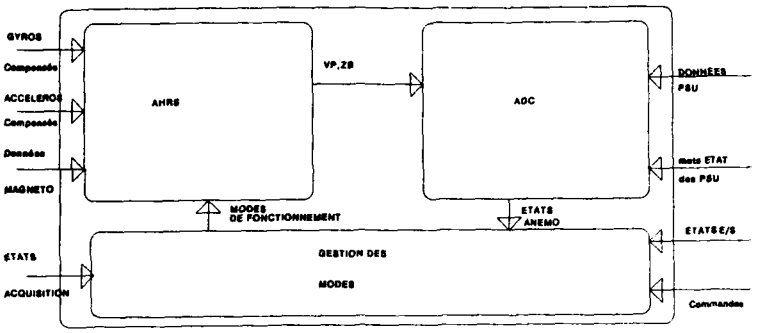
## CYCLE DE VIE



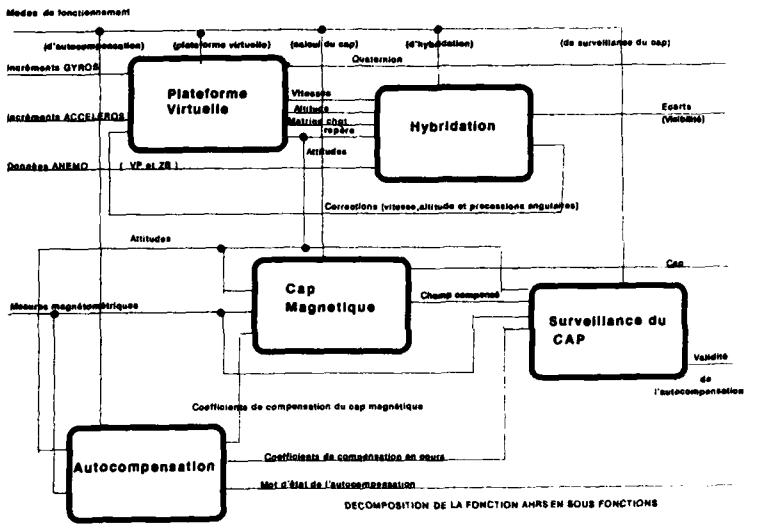




SYNOPTIQUE SIMPLIFIE DU SYSTEME SRP

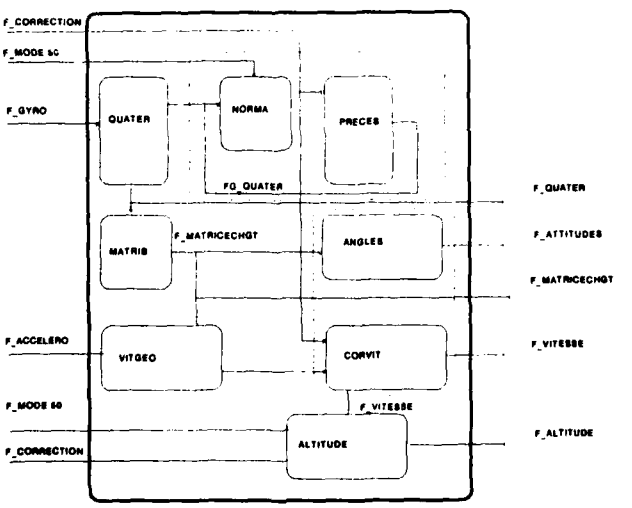
ANNEXE 2




F.D.C. (Flight data Computer) vision Fonctionnelle de la carte unité de traitement




DECOMPOSITION DE LA FONCTION AHRS EN SOUS FONCTIONS




DECOMPOSITION DE LA SOUS FONCTION PLATEFORME VIRTUELLE EN SEQUENCES ET MODULES

# FAULT TOLERANCE VIA FAULT AVOIDANCE

B D Bramson
RSRE, Malvern, Worcs, WR14 3PS, UK



**Abstract**

A safety-critical system is proposed whose architecture is based upon software components that have diverse specifications and diverse implementations. It is claimed that a proof of correctness of one of the components implies a proof of safety of the system. The claim is illustrated using the MALPAS Intermediate Language as a design language and Compliance Analysis as a verification technique.

## 1 INTRODUCTION

I want to propose quite generally a software system whose design is guaranteed to satisfy some specified property even though the system may not have been submitted in its entirety to a formal correctness proof.

By a **software system** in the context of this conference I mean the instructions of some programmable machine that receives sensed inputs and delivers outputs either to control some piece of equipment or to provide advice to the human user. By the **design** of the software I envisage some high-level representation of the machine's behaviour sufficiently expressive for us to be able to reason about some of its properties. Both the implemented software and its design may make use of sequential logic, as with standard programming languages, but the **specified property** will be expressed in a form that is quite distinct, being closer to parallel logic; and if the specified property is related to safety then we will need to address such issues as ambiguity, inconsistency and completeness. It follows that the language of mathematics will be required although there is no reason for this to be a bar to the creative engineer. History abounds with examples where the development of engineering and mathematics proceed hand in hand. It turns out in fact that propositional calculus is of great value when it comes to specifying computer programs.

I began by referring to a **formal correctness proof** which was intended to imply a mathematical comparison of the software with its specification. (Of course there are various styles and various degrees of formality.) Several systems world-wide, notably Gypsy [1], SPADE [2] and MALPAS [3] automate the proof of correctness. In each case, a given program together with its specification is submitted to a suite of programs that performs program verification via a process of static analysis. With Gypsy and SPADE, the process is performed in two stages. First, verification conditions are produced, namely theorems to be proved true, while the second stage comprises a theorem prover. MALPAS is slightly

different in so far as it employs **Compliance Analysis**, a technique that amounts to a revelation of incorrectness. The absence of revelation implies proof of correctness!

The reader may begin to wonder why a paper on program proving, indeed fault avoidance, should appear in a conference on fault tolerance where the principles of replication and redundancy have held sway for many years. The reason is that I do not believe that the traditional methods of redundant hardware solve anything by themselves when it comes to system design. This is true even of the design of a piece of hardware containing a loom of wiring. For if the wiring diagram is wrong and if the wireman follows the wiring diagram then you have the potential for common-mode failure which no amount of replication will rectify. Even if you employ separate contractors to fabricate different implementations of the device, the problem will persist if the same wiring diagram is used.

In the world of software a similar state of affairs exists. Identical microprocessors running identical software in parallel channels will contain common software implementation errors. Diverse microprocessors running diverse implementations but written to the same design will contain common design errors. Most importantly, a specification that is ambiguous, inconsistent or incomplete is likely to create common problems in all channels. Quite apart from that, the implementation of diverse programs to a common specification is a highly expensive process! Indeed, because of the issues of learning curves and fixed budgets it can be more cost-effective to focus one's resources on a single high-quality program.

However, there could be systems where a combination of diversity and formal proof offers the ideal solution. I am thinking of safety-monitors and what a colleague, W J Cullyer, once described as the "Get me home" program. The idea is that in parallel with the main, possibly untrusted process runs a smaller, standby process with weaker function that nonetheless has been proven correct with regard to some vital property. When both processes have run, a safety-monitor executes a dynamic assertion to check whether the result delivered by the main process is safe and suitable: if *not*, the output from the standby process is employed.

Section 2 describes the Compliance Analysis of software using simple mathematics based on the theory of sets. When a piece of software is compared with its specification the dangerous inputs to the software are displayed in algebraic form.

Section 3 presents a system that comprises a main process, a standby process and a safety-monitor. The system is expressed both diagrammatically in data-flow style and more formally using the MALPAS Intermediate Language [4]. It turns out that proofs of correctness of the standby process and of the safety-monitor together imply that the system as a whole satisfies the required safety property.

## 2 COMPLIANCE ANALYSIS OF SOFTWARE

Figure 1 describes a finite machine, with no internal states, that receives inputs $IN$ and delivers outputs $OUT$. A certain property of the machine's intended behaviour is specified by the pair $(PRE, POST)$. $PRE$ is a subset of $IN$ and defines those inputs for which the machine is intended. $POST$ is a subset of the Cartesian product $IN \times OUT$ and defines a relation (shaded) from inputs to outputs which the machine is required to satisfy. Thus whenever the input *in* lies in $PRE$, the output *out* must be such that the pair $(in, out)$ lies in $POST$. Inputs outside $PRE$ are illegal and nothing is then said about the machine's

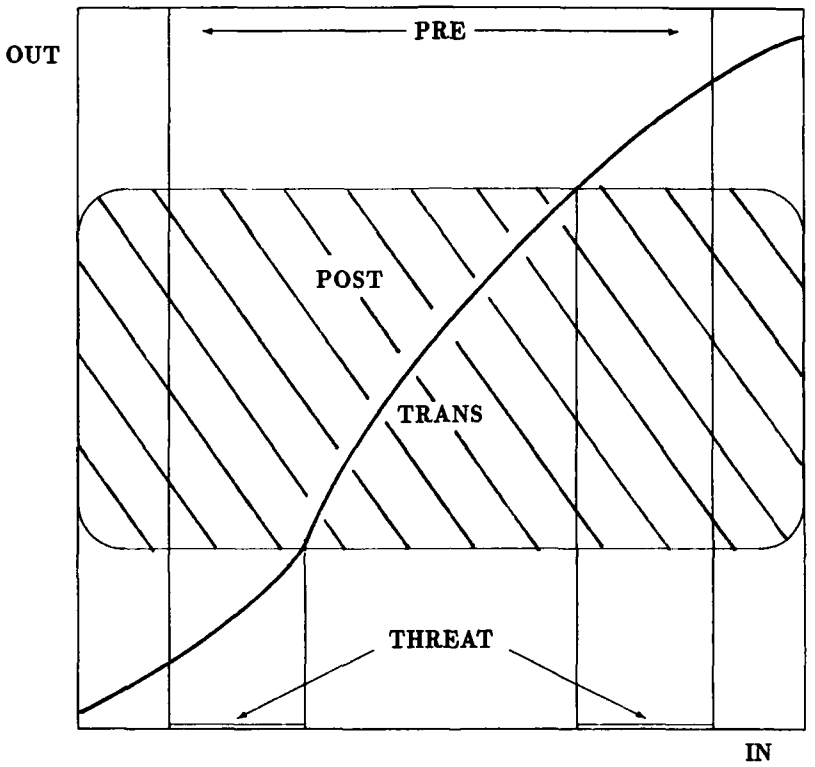



Figure 1: The compliance analysis of a finite machine with no internal state. $THREAT$ comprises those inputs which, on running the machine, lead to outputs violating the specification.

behaviour.

Specifications can be deterministic. For example the designer of a voltage amplifier might demand that it be suitable for inputs between 1 and 10 volts and that the output be ten times the input. (For an expansion see [5].) However, specifications need not be deterministic. "Whenever the input exceeds 9 volts, the red light comes on" is non-deterministic because it says nothing about other outputs.

In figure 1, the specification is non-deterministic: for a given input lying in $PRE$ many outputs would satisfy $POST$. However, this particular machine is deterministic and always terminates, mapping $IN$ into $OUT$ by means of the function $TRANS$.

Compliance analysis compares the implementation $TRANS$ against the specification $(PRE, POST)$ by calculating those inputs, $THREAT$, that lie in $PRE$ but lead to outputs that violate $POST$. Specifically, $THREAT$ is defined via a relation $RISK$ according to

$$RISK = TRANS - POST, \quad (1)$$

$$THREAT = PRE \cap domain(RISK), \quad (2)$$

where the domain of $RISK$ is its projection onto $IN$. Thus $THREAT$ comprises those inputs to the machine considered to be dangerous; which is clearly useful to compute. Some readers may note that the domain of $RISK$ is just the complement (negation) of Dijkstra's weakest pre-condition. (I must thank C A R Hoare for pointing this out to me.) Indeed,

$$THREAT = \emptyset \text{ if and only if } ((PRE \times OUT) \cap TRANS) \subseteq POST. \quad (3)$$

For a machine implemented in software, $IN$ and $OUT$ comprise the states of memory locations respectively before and after the execution of $TRANS$, together with sequences of data input and output. $TRANS$ represents some program intended to satisfy the post-condition $POST$ given the pre-condition $PRE$.

# 3 DIVERSE ARCHITECTURE

The data flow diagram presented in figure 2 depicts the design of a process, the *SAFETY-FILTER*, that generates a sequence of outputs from a sequence of inputs in such a way that each input-output pair is guaranteed to satisfy some specified property relating to safety.

*SAFETY-FILTER* comprises three processes:

*MAIN* generates a sequence of outputs from a sequence of inputs. In order to allow for the possibility that *MAIN* might generate erroneous outputs, *MONITOR* and *STANDBY* are included.

*STANDBY* also generates a sequence of outputs from a sequence of inputs but each input-output pair is guaranteed to satisfy the specified property. However, *STANDBY* performs fewer functions than *MAIN* being simpler in design.

*MONITOR* is also highly trusted, its purpose being to check each output from *MAIN* against each input with regard to the required property. If *MAIN* performs safely and if *ERROR-COUNT* lies below some given threshold, then *PROPOSAL* is output; otherwise the alternative output from *STANDBY* is employed.

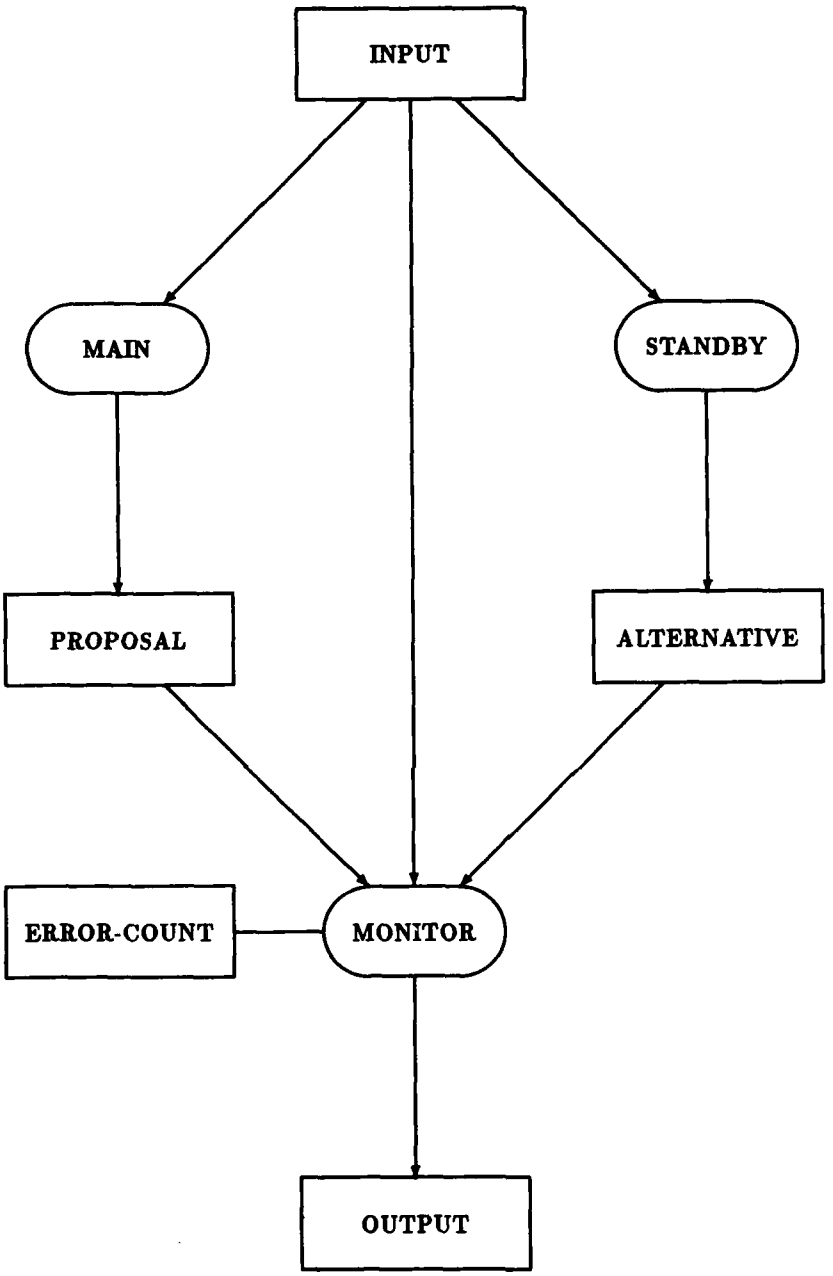



**Figure 2: THE SAFETY-FILTER receives a sequence of inputs and delivers a sequence of outputs. Ovals indicate processes while rectangles depict data.**

My purpose now is to display a piece of MALPAS Intermediate Language (IL) that represents part of the design of the safety-filter. IL is a strongly typed language that may be used to model either the design or the implementation of some software system so that we may reason about its properties using methods of automatic static analysis. However, the reader should note that in the present context IL is used as a design language rather than as a model of some programming language.

Before delving into detail it will be worth saying a few words about the nature of IL itself. An IL program comprises a declarative part and an algorithmic part. Types, functions and operators may be declared without elaboration. They may also be assigned semantic meanings by means of records and rewrite rules while rules for manipulating integers and Booleans are in-built. Parametric types are also supported so that generic packages, for example for lists and arrays, may be defined.

A procedure may be declared without elaborating its body, it being sufficient merely to list its formal parameters together with their types and classes (IN, INOUT, OUT). The parameter passing mechanism is that of "copy in, copy out". However, a procedure to be called by another procedure or by itself requires a more detailed specification. At the very least, this must be a relation from inputs to outputs detailing the data dependency.

The algorithmic part of an IL program contains the procedure bodies. Within each body, local variables may be declared at the outermost level. Sequential and parallel assignment are both supported. Procedures may be called, in series or in parallel, but their specifications are executed rather than their bodies. This means that recursion may be modelled. Standard control structures, IF..THEN..ELSE..ENDIF, LOOP..ENDLOOP, are all supported.

More details of all this may be found elsewhere [4]. For brevity I shall merely highlight those features relevant to the design in question. The numbers that follow relate to figures 3 and 4.

1. At the design level, *input* and *output* are abstract types.
2. A *FUNCTION* in IL is a mathematical function rather than a typed procedure.
3. The type-extension *list* was declared in a preamble of standard declarations.
4. Rewrite rules, heralded by *REPLACE*, assign meanings to functions. In this example, the rule is recursive.
5. In the preamble, *empty* was declared as an "untyped constant". The type follows the colon.
6. For non-empty lists, *FIRST* and *REST* project out the first element and remainder respectively.
7. The declaration of *filter* defines a black box that receives inputs and delivers outputs.
8. Compliance analysis will compare the body of filter (see later) against the specified *POST*-condition. The $'$ refers to the initial state.
9. The *DERIVES* relation says merely that the final state of $y$ is some function of the initial state of $x$. When *main* is called by *filter* this information is used.

```
     TITLE filter;

[1]  TYPE input, output;

[2]  FUNCTION untrusted(input-list, output-list): boolean;
     FUNCTION safe(input, output): boolean;
[3]  FUNCTION all_safe(input-list, output-list): boolean;
     CONST threshold: integer;

[4]  REPLACE (x: input-list; y: output-list)
        all_safe(x, y)
[5]  BY y = empty:output-list IF x = empty:input-list,
[6]  BY safe(FIRST x, FIRST y) AND all_safe(REST x, REST y) AND
        NOT(y = empty:output-list) IF NOT(x = empty:input-list);

[7]  PROCSPEC filter (IN in: input-list
                      OUT out : output-list)
[8]  POST  all_safe('in, out);

     PROCSPEC main (IN x: input-list
                    OUT y: output-list)
[9]  DERIVES y FROM x
     POST    untrusted('x, y);

     PROCSPEC standby (IN x: input-list
                       OUT y: output-list)
     DERIVES y FROM x
     POST    all_safe('x, y);

     PROCSPEC monitor (IN in: input-list
                       IN prop: output-list
                       IN alt: output-list
                       INOUT err: integer
                       OUT out: output-list)
     DERIVES out FROM in & prop & alt & err,
             err FROM in & prop & err
[10] PRE     all_safe(in, alt)
     POST    all_safe('in, out);
```

Figure 3: DESIGN OF SAFETY-FILTER: declarative part

```
[11] PROC filter;
[12] VAR proposal, alternative: output-list;
     VAR error_count: integer;
       error_count := 0;
[13]   MAP
       main(in, proposal);
[14]   standby(in, alternative) ASSUME POST
       ENDMAP;
[15]   monitor(in, proposal, alternative, error_count, out) ASSUME POST
     ENDPROC;

[16] PROC monitor;
     VAR temp: output;
       IF in = empty:input-list
         THEN out := empty:output-list
[17]   ELSIF err > threshold OR prop = empty:output-list
         THEN out := alt
         ELSE IF safe(FIRST in, FIRST prop)
                THEN temp := FIRST prop
                ELSE err := err + 1;
                     temp := FIRST alt
              ENDIF;
[18]          monitor(REST in, REST prop, REST alt, err, out) ASSUME POST;
[19]          out := L temp @ out
       ENDIF
      ENDPROC

FINISH
```

Figure 4: DESIGN OF SAFETY-FILTER: algorithmic part for FILTER and MONITOR

10. When *main* calls *monitor* the analyser checks that the *PRE*-condition is satisfied. (The integrity of *monitor* depends upon satisfaction of the pre-condition.)

11. This is the body of *filter*.

12. *VAR* declares a local variable.

13. *MAP* and *ENDMAP* embrace parallel logic. Thus *main* and *standby* are called in parallel. The syntax of IL requires that any piece of data is written to at most once within a *MAP..ENDMAP* construct.

14. *ASSUME POST* means that we are using the post-condition of *standby*. This is needed to prove the pre-condition for *monitor*.

15. The post-condition for *monitor* is needed to prove the post-condition for *filter*.

16. It turns out that the correctness of *monitor* with regard to the specified post-condition does not depend on the precise details of its body. Many designs would meet the post-condition.

17. An empty *prop* with a non-empty *in* might arise if *main* failed to write to the output list for certain inputs.

18. Thus at the level of the design *monitor* calls itself recursively. If the original input list had length $n$ the list here has length $n-1$. Thus *ASSUME POST* here is a step in a proof by induction.

19. *L* constructs a list with a single element while @ is the concatenation operator.

Finally, it is worth remarking on the model used here for a real-time system. Each process has been represented by an IL procedure that operates on lists of values. In a sense each list is envisaged as a single data item so that, at least for the current level of description, *main*, *standby* and *monitor* are called precisely once by *filter*. Moreover, the call to *monitor* follows those to *main* and *standby*, the latter occurring in parallel. That of course is not to say that in the ultimate implementation *monitor* waits for *main* and *standby* to terminate before commencing itself!

# 4 CONCLUSION

The purpose of this paper has been to present a hypothetical processing system designed to satisfy some specified property even though only a small part of it has been submitted to a formal proof of correctness.

However, there is an important practical point that I have ignored. For nothing has been said of the range of problems to which the concept of software monitoring and diversity is applicable. Do applications exist? In the example given have we any grounds for believing that the task of proving the correctness of *STANDBY* and *MONITOR* is any easier than that of proving *MAIN*?

In fact there is a deeper issue concerning the underlying philosophy and with which the reader will be right to feel a sense of unease. For it is one thing to envisage the

possibility of hardware faults, and I definitely exclude logical errors of design from these, but it is quite another to accept the existence of software errors. And yet these are a fact of life for a variety of different reasons, not the least being fecklessness. In the long term surely our efforts must be directed towards methods for producing large systems with the proof of correctness in-built? Such methods must be mathematically based, cost-effective and usable and they will have to relate to the separate issues of specification, design and implementation. They will also have to allow for the problems of communication between engineer, mathematician and computer scientist. Large systems will have to be decomposed into smaller sub-systems, an approach familiar to engineers, with the smaller parts either performing in parallel or related via a hierarchy of descriptive levels.

Finally, we will need to account for what I shall term *The Principle of Software Uncertainty.* Roughly speaking, it may be stated as follows:

- Before development commences,
  the customer does not know in detail what he wants.

## References

[1] D I Good
Mechanical proofs about computer programs
Phil Trans R Soc Lond A **312** 389-409, 1984.

[2] SPADE
Program Validation Ltd, Southampton, Hampshire, UK.

[3] B D Bramson
Tools for the specification, design, analysis and verification of software
RSRE report 87005, 1987 .

[4] MALPAS Intermediate Language Manual
RTP Software Ltd, Farnham, Surrey, UK.

[5] C A R Hoare
Programs are predicates
Phil Trans R Soc Lond A **312** 475-489, 1984.

# PILOTED SIMULATION VERIFICATION OF A CONTROL RECONFIGURATION STRATEGY FOR A FIGHTER AIRCRAFT UNDER IMPAIRMENTS*

by

**Richard Mercadante** (Sr Flight Controls Engineer)
Grumman Corporation
Aircraft Systems Division
Bethpage, NY 11714-3582
United States

## SUMMARY

Piloted simulation performed at the USAF large amplitude multi-mode aerospace research simulator (LAMARS) verified the capability of a reconfiguration strategy to improve aircraft controlability. USAF Tactical Air Command pilots and test pilots from a number of organizations evaluated the characteristics of a next-generation fighter aircraft subjected to control surface damage and/or actuation failures. Tests were performed both with and without the aid of the reconfiguration strategy. For the aircraft configuration simulated, pilot opinions, ratings, and target tracking scores demonstrated the capability of the system to improve aircraft response for a large variety of control surface impairments throughout the subsonic flight envelope. Results ranged from slight to dramatic improvement and departure prevention.

## NOMENCLATURE

| | |
|---|---|
| AAD | Automatic Alert Display |
| ACLS | Automatic Carrier Landing System |
| ACM | Air Combat Maneuver |
| AGL | Above Ground Level |
| CAF | Canadian Air Force |
| C-H | Cooper-Harper rating |
| CRCA | Control Reconfigurable Combat Aircraft |
| CSS | Control System Status |
| DOF | Degrees of Freedom |
| FCC | Flight Control Computer |
| FCMDS | Flight Control Maintenance Diagnostics System |
| FCS | Flight Control System |
| FDIE | Failure Detection, Isolation, and Estimation |
| HQ TAC | Headquarters, Tactical Air Command |
| LAMARS | Large Amplitude Multi-mode Aerospace Research Simulator |
| LVDT | Linear Variable Data Transducer |
| HUD | Head-Up Display |
| MFD | Multi-Function Display |
| MTBF | Mean-Time Between Failures |
| NASA | National Aeronautics and Space Administration |
| PPA | Positive Pilot Alert |
| PSR | Psuedo-Surface Resolver |
| RCL | Reconfigurable Control Laws |
| RFC | Research Fighter Configuration |
| RS | Reconfiguration Strategy |

*This work was performed under U.S. Air Force contract No. F33615-84-C-3607.

| | |
|---|---|
| RTR | Real-Time Reconfiguration |
| SRFCS | Self-Repairing Flight Control System |
| STOL | Short Takeoff/Landing |
| TE | Trailing Edge |
| TF/TA | Terrain Following/Terrain Avoidance |
| TPS | Test Pilot School |
| USAF | United States Air Force |
| WRDC/FIGL | Wright Research and Development Center/Flight Dynamics Laboratory |
| WRDC/FIGX | Wright Research and Development Center/Flight Dynamics Laboratory Advanced Programs Office |

## 1 - INTRODUCTION

Over recent years, air warfare scenarios have indicated a need to maintain technically superior NATO aircraft to offset a numerically superior threat. Previous studies (Ref 1 through 3) have addressed this threat by directing efforts toward increased aircraft survivability. In particular, the problem of aircraft control under flight control system (FCS) failure and ballistic damage was deliberated. As a result, methods of reconfiguring flight control laws were developed and other FCS areas with potential for improvement were highlighted. Specifically, increased reliability and maintainability and pilot notification were identified.

The self-repairing flight control system (SRFCS) program was developed to expand upon these efforts. This advanced development effort, sponsored by the Flight Dynamics Laboratory at the Wright Research and Development Center (WRDC/FIG), was oriented toward reducing life cycle cost (LCC) of current and future aircraft by increasing mean-time between failure (MTBF), decreasing aircraft weight and complexity, and improving survivability. To achieve this goal, the SRFCS Program targeted two areas of flight control system operation for study.

One study was chartered with increasing MTBF by reducing the time and manpower required to diagnose reports of in-flight FCS problems. This flight control system maintenance diagnostics study (FCMDS) evaluated, developed, and tested both on-board and ground-based maintenance diagnostics systems. The other area of study targeted the FCS design itself. The control reconfigurable combat aircraft (CRCA) study was chartered with evaluating current aircraft configurations and FCS design practices, simplifying FCS complexity, and developing a reconfigurable FCS to extend combat persistence in the event of control effector damage or actuation failures. During the course of the CRCA study, FCS complexity was reduced (Ref 4), a reconfiguration strategy was developed (Ref 5 through 7), and the resulting aircraft/control law configuration was evaluated (Ref 8).

This paper presents the results obtained during piloted evaluation of the CRCA performed at the USAF LAMARS facility.

## 2 - DISCUSSION

### 2.1 CONFIGURATION DESCRIPTION

The Grumman/NASA Research Fighter Configuration (RFC), an aft-swept wing, close-coupled canard, air superiority fighter configuration (shown in Fig. 1), served as the baseline aircraft for the CRCA study. As a result of preliminary study efforts, the original RFC was modified. In particular, the canards were installed at a 30-degree dihedral angle to increase directional power resulting from differential deflections. This was done to provide yaw axis redundancy in the event of rudder impairment (single vertical tail). In addition, the wing's trailing edge was changed, from four separate devices per side to three, to reduce actuator count. Thrust vectoring capability was

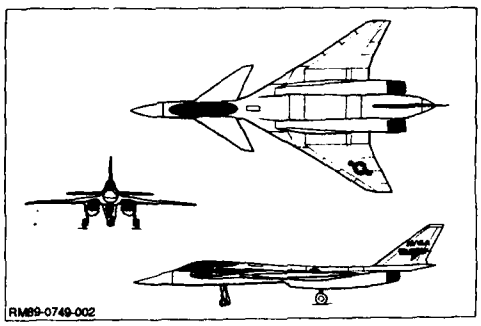



Fig. 1 Grumman/NASA Research Fighter Configuration

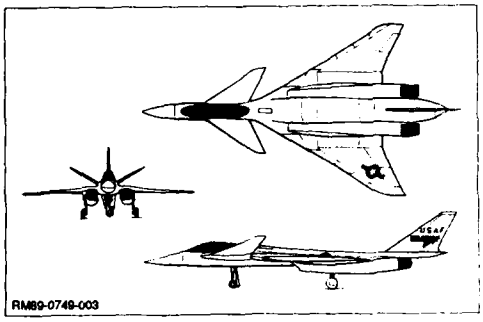



Fig. 2 Control Reconfigurable Combat Aircraft (CRCA)

eliminated to diminish actuation requirements. The result is the control reconfigurable combat aircraft (CRCA) shown in Fig. 2.

Aerodynamic data describing the original RFC were obtained in the NASA/Langley 16 ft (4.88m) wind tunnel. Modifications reflecting the CRCA were tested in Grumman's Low Speed Wind Tunnel, and the data base modified. A 6-degree-of-freedom digital simulation was constructed using these data, along with models of F-16 Integrated Servo Actuators (ISAs), kinematic sensors, two Pratt & Whitney F-100 engine models (modified to required thrust), and a set of nominal control laws (for simulation verification). A block diagram of the simulation is given in Fig. 3.

The CRCA's nine primary control surfaces were driven by simplex actuators, except for the rudder, which used a dual-tandem actuator. Both canards and all six wing trailing edge devices (two elevators and four flaperons) were simulated by fourth-order models of simplex F-16 ISAs. The rudder was driven by a dual-tandem version of these actuators. Each hydraulic system powered one canard, one elevator, and two flaperon actuators, as well as one chamber of the rudder actuator. Switch valves were minimized for MTBF reasons. The resulting hydraulic system arrangement is shown in Fig. 4. Failure of any one hydraulic pump caused loss of power to four simplex actuators.

Longitudinal control of the CRCA was provided by a g-command system in up-and-away flight, and a pitch rate command system in the STOL mode. In all cases, canards and wing trailing edge devices were driven with a unity gain. Lateral control was provided via a stability axis roll rate command system with differential deflection of the wing trailing edge devices. The directional system was a simple command system that derives its power from rudder and differential deflection of the canards, except at high dynamic pressure, where only rudder was used because surplus power existed. A block diagram of the complete control system is shown in Fig. 5.

## 2.2 DAMAGE & FAILURE MODELING

Each control surface was modeled separately within the aerodynamic data base. Control surface damage was modeled in terms of "percent effectiveness loss" of total control power, instead of area loss. This was consistent with the FDIE's operation, which was concerned with locating the impaired surface and estimating the effects of the damage.

Static and dynamic canard-wing-body-tail characteristics were modified to reflect the loss of each control surface's contribution to these derivatives. The static lateral/directional effects of one canard were modeled separately within the aerodynamic

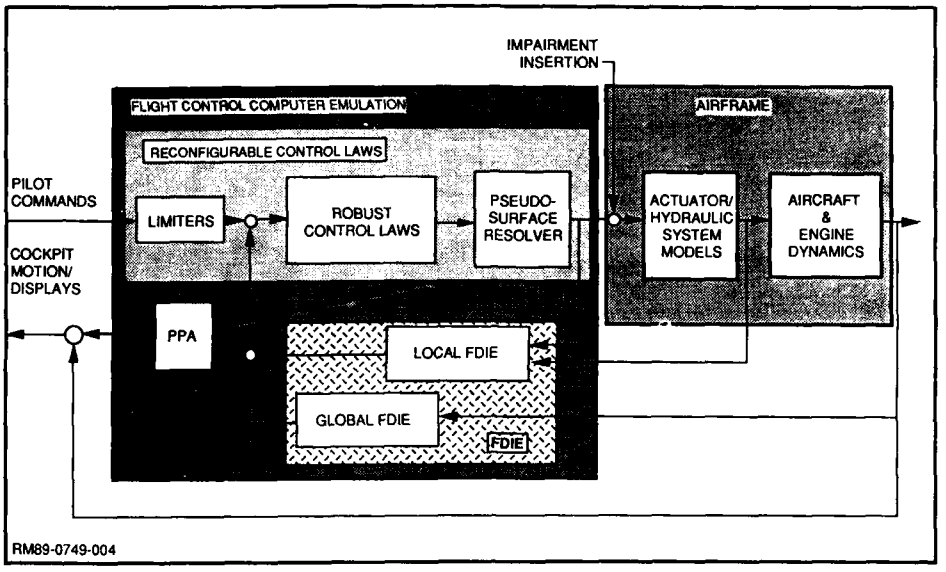



**Fig. 3 CRCA Simulation Block Diagram**

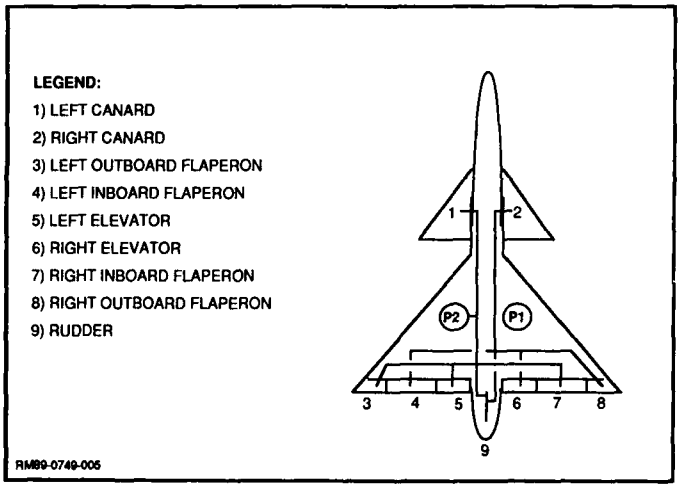



**Fig. 4 CRCA Hydraulic System Arrangement**

database. These data were generated in the wind tunnel by removing one canard. Therefore, the change in forebody pressure distribution and local wing angle-of-attack were taken into account, as well as control power loss.

A separate subroutine simulated actuation failures so damage to, and loss of control of a surface could be simultaneously simulated. The actuation failures included locked, runaway, and floating; partial or total hydraulic system failure scenarios were also considered. In the event of a floating actuator, the selected surface follows a model of its trailing position. Canards and wing trailing edge devices float with angle-of-attack, while the rudder floats as a function of sideslip angle. The hydraulic system was ground-ruled to have its own simple FDI. In the event of loss of hydraulic system pressure, the FDI commanded the simplex actuators into a damped by-pass mode. This mode results in a heavily damped fail-to-trail position.

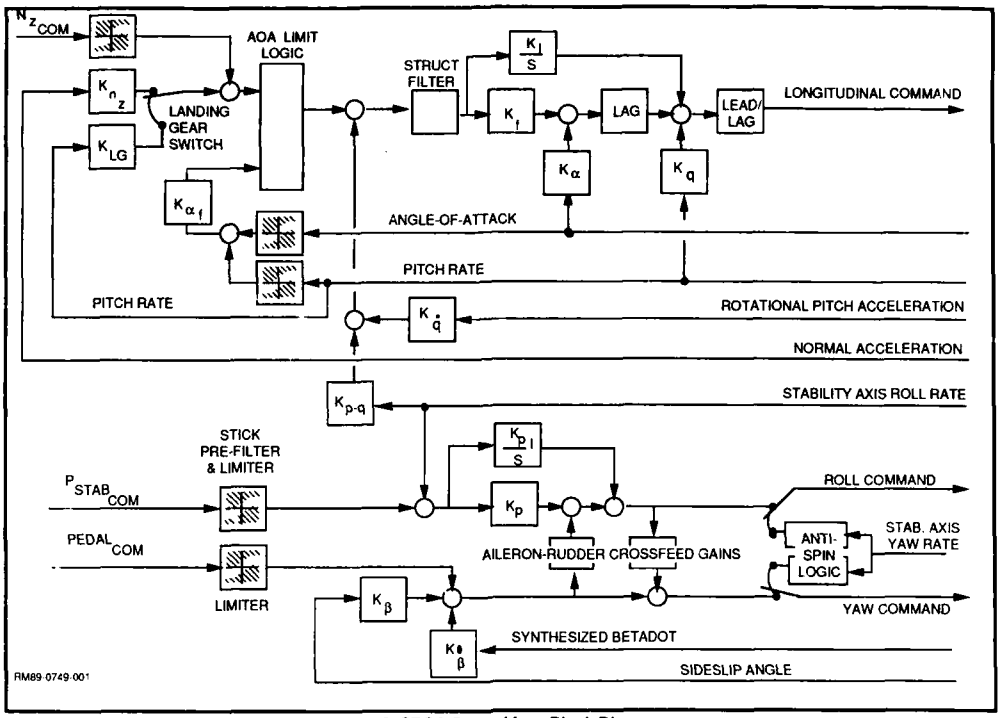



Fig. 5 CRCA Control Law Block Diagram

## 2.3 RECONFIGURATION STRATEGY

The reconfiguration strategy (RS) is composed of a robust control law coupled with a pseudo-surface resolver (PSR) forming a reconfigurable control law (RCL), a failure detection, isolation, and estimation (FDIE) algorithm, and a positive pilot alert (PPA) system. The result of this RS is the capability for real-time reconfiguration for a variety of single, simultaneous and sequential control surface impairment scenarios.

Current redundancy management schemes (Ref 9 and 10) contain much of the technology required to construct the necessary actuator failure, detection, and isolation algorithms. The challenge posed for reconfiguration was to develop reconfigurable control laws that take advantage of this information, provide adequate aircraft performance, and allow reduction in component complexity.

Once the reconfigurable control laws were designed for actuation failures, the next logical step was extension of the technology to loss of control surface effectiveness to battle damage, thereby increasing survivability. This required a damage detection, isolation, and estimation scheme. The challenge here was to design an FDIE algorithm that could isolate and estimate damage effects within time and accuracy constraints for safety of flight, as well as those imposed by onboard computer resources.

To address these criteria, four subsonic flight conditions were selected for design and development of the reconfiguration strategy. These design points, selected for their unique characteristics, are:

- Air Combat Maneuver (ACM) Entry - 0.900 Mach, 30,000 ft (9,144m)
- ACM Exit - 0.275 Mach, 10,000 ft (3,048m)
- Terrain Following/Terrain Avoidance - 0.900 Mach, sea level
- Short Takeoff/Landing (STOL) - 0.200 Mach, sea level.

The corresponding characteristics are high maneuverability, high angle-of-attack and low dynamic pressure, maximum dynamic pressure, and low dynamic pressure, respectively.

General design goals established in the Grumman-developed test plan tasked the RS with: returning the aircraft to its original state for single flaperon impairments; providing MIL-F-8785C-defined Level 2 flying qualities for multiple flaperon, single canard or rudder impairments; and Level 3 to Level 2 flying/handling qualities for partial or total hydraulic system impairments. The pseudo-surface resolver strived to reach these goals by reducing cross-coupling of single-axis inputs, restoring aircraft response to the pilot, and restoring damping.

Appended to the forward path of the robust control laws was a pseudo-surface resolver (PSR). The robust control laws provided departure resistance under impairment and time for the failure detection, isolation, and estimation (FDIE) algorithms to operate. Once the FDIE had determined the existence of an impairment, isolated it as to type, and estimated the magnitude in the event of damage, the PSR was notified. The PSR attempted to restore the original aircraft control capability through use of a pseudo-inverse matrix calculation, while reducing cross-coupling resulting from the impairment.

The FDIE system was broken down into two distinct algorithms: a local FDIE and a global FDIE. The local FDIE was used to determine and isolate impairments local to the actuator using information from the linear variable data transducers (LVDTs) and surface commands. The global FDIE operated on information of global magnitude with respect to the aircraft. Aircraft state information (e.g., acceleration, rates, and attitudes) were used along with pilot commands, flight control computer (FCC) commands, and local FDIE information to estimate control surface change in effectiveness.

The positive pilot alert (PPA) system used information determined within the FDIE and action taken by the reconfigurable control Laws. Aircraft limits, flight status, and estimates of control authority available were determined for both current aircraft state and landing.

The PPA was designed to be generic to next-generation fighters, but its development was tailored for the CRCA's evaluation. The standard PPA head-up display (HUD) is shown in Fig. 6. Analog displays of angle-of-attack, normal load factor, and airspeed were

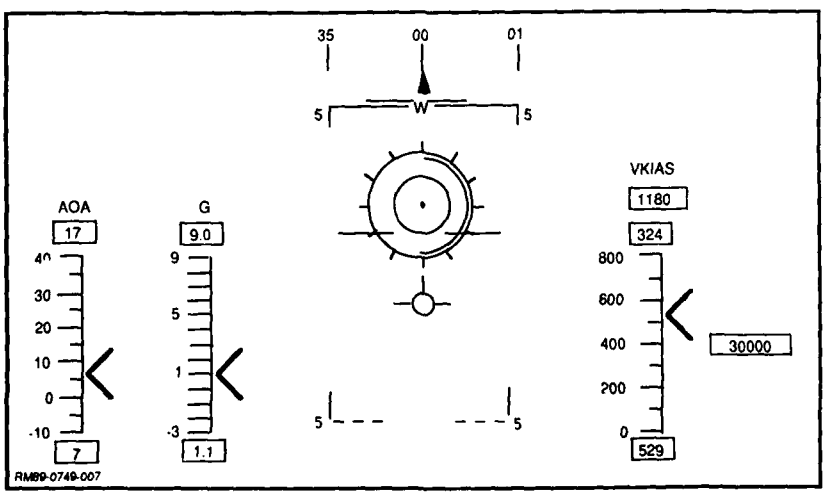



**Fig. 6 Standard PPA HUD**

provided. Maximum limits of each parameter were displayed via digital read-out in the boxes above the tapes, as well as the minimum limit in the case of airspeed. In addition, a flashing indicator appeared on the analog display when the current value of a parameter was 80% or more of its limit (the minimum velocity indicator appeared when the airspeed was less than or equal to 120% of the minimum velocity). The current value of each parameter was digitally displayed below each tape.

When an impairment was detected and isolated by the FDIE, an automatic alert display (AAD) appeared on the HUD (Fig. 7). The AAD notified the pilot of impairment severity, system impaired, current mission status, and maneuver capability. This same AAD appeared in the upper left-hand corner of the color, multi-function display (MFD) with one modification. The impairment severity message was deleted to conserve area and the remaining messages were color-coded: red for warning, amber for caution, and green for normal.

At this point, the pilot could select the "acknowledge" button, clearing the AAD from both locations and either continue with his other tasks or select "PPA." The first PPA menu to appear was the flight status display (Fig. 8). It indicated flight status for each mode (color-coded), mission mode status, further elaboration on maneuver restrictions, control power available, and aircraft limits. (Aircraft limit labels also flashed when limits were approached.) The default setting for this menu caused current aircraft capability to be displayed. Selection of "LAND" (lower left side) provided estimates of controllability in the approach mode.

Pressing the bezel labelled "FCS" produced the control system status display (Fig. 9). The key feature of this display is an exaggerated planform view of the aircraft with the impaired surface(s) colored red and labeled as to impairment type. In the case of damaged surfaces, the percent effectiveness lost was provided, and, in the case of a locked surface, its position was given; hydraulic pressure was also presented.

Figure 10 shows the emergency procedures display. This provides pertinent information for either current aircraft state or the landing mode.

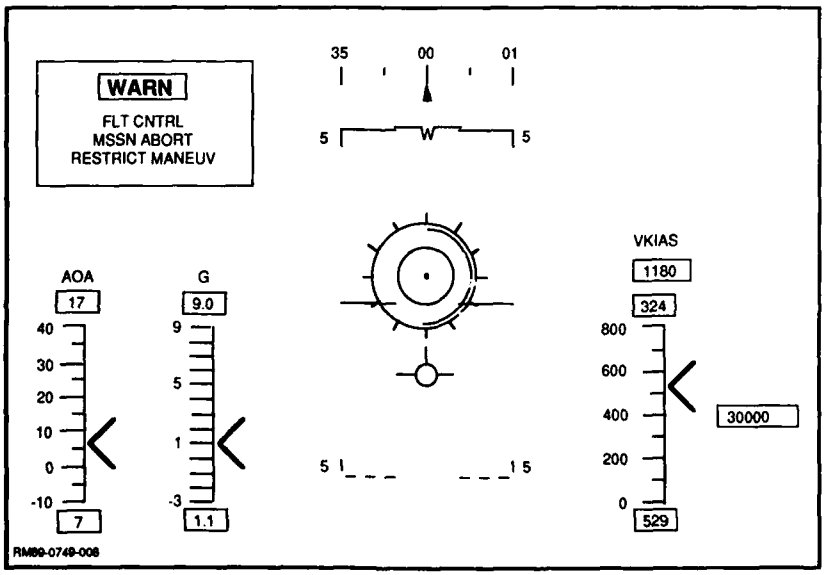



**Fig. 7 AAD On HUD**

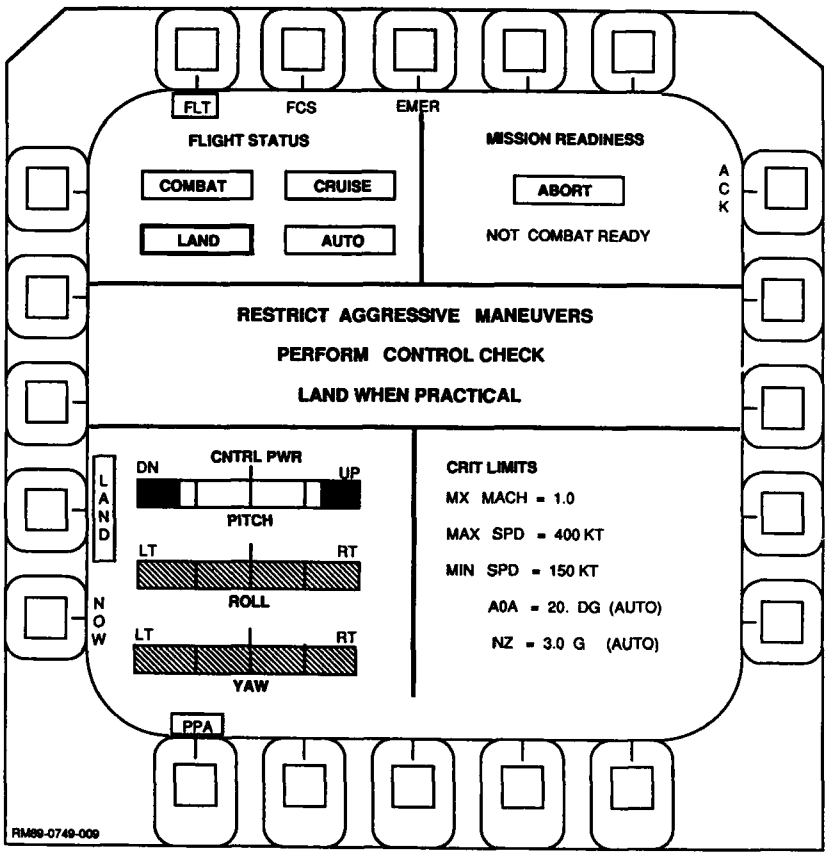



**Fig. 8 Flight Status Display**

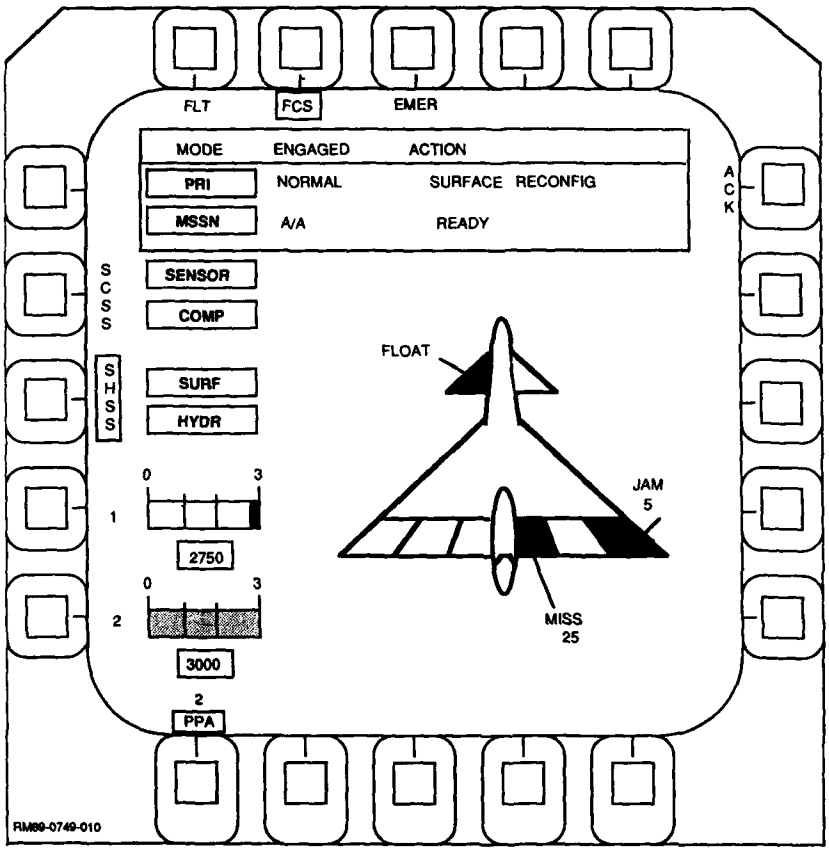



**Fig. 9 Control System Status Display**

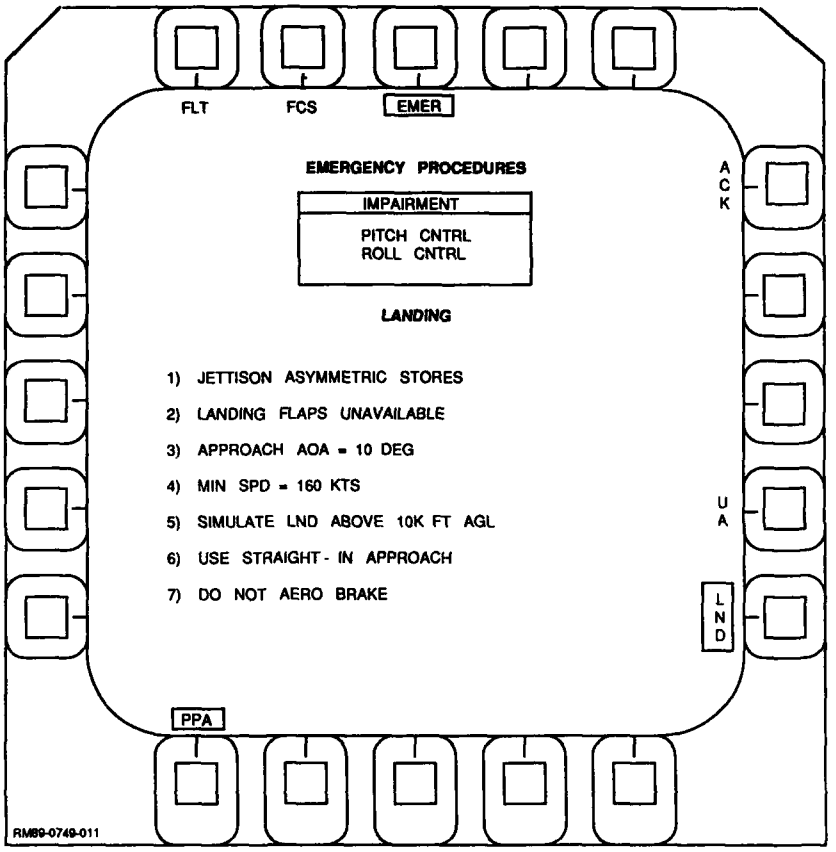



**Fig. 10 Emergency Procedures Display**

## 2.4 TEST CONDITIONS

Four pilots participated in the 23 January through 3 February 1989 verification of the reconfiguration strategy's operation. Two pilots were from the USAF Tactical Air Command's Headquarters (HQ TAC), one was a test pilot from the USAF Test Pilot School (TPS) faculty, and one was a NASA/Dryden test pilot. Each had more than 4,500 hours of jet experience.

Following pilot familiarization flight time, the pilots were asked to perform a variety of open- and closed-loop maneuvers and tracking tasks. These included ACM target tracking against a previously stored maneuvering CRCA, STOL approaches over a 1,500:1 scale terrain board, and TF/TA waypoint following at 350 ft above ground level (AGL) over a 5,000:1 scale terrain board. Reconfiguration strategy performance was graded against the impaired/non-reconfigured CRCA using Cooper-Harper ratings (Ref 12), target tracking scoring, and pilot workload measurement for both the ACM and STOL tasks. All tests were flown with simulation motion.

The Cooper-Harper rating scale (Fig. 11) was used to grade handling qualities of open- and closed-loop maneuvers and tracking tasks. Initial verification tests were performed with the PPA disabled to avoid introducing any bias into pilot ratings by providing knowledge of the impairments and control law state (i.e., reconfigured or not). The PPA was later enabled and a selected subset of aircraft impairments was reflown.

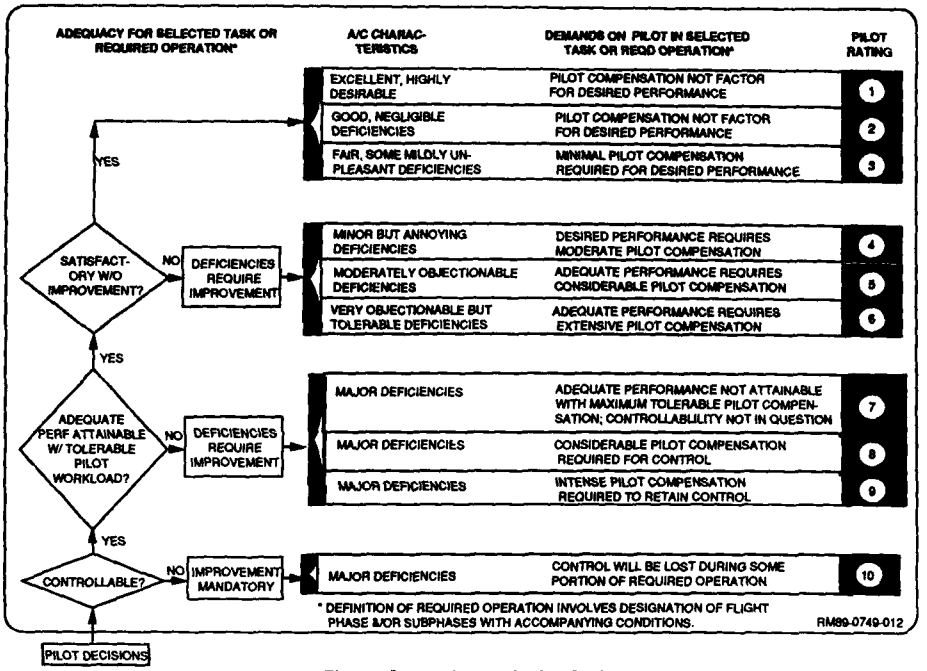



**Fig. 11 Cooper-Harper Rating Scale**

The RS was tested for the impairments shown in Fig. 12. Impairment No. 18 was a damage scenario whereby the inboard flaperon was completely lost to a ballistic hit, as well as the hydraulic line supplying the outboard flaperon. Its loss of pressure was sensed and the actuator placed into the damped by-pass mode. In Impairment No. 19, the left canard actuator suffered a catastrophic leak causing it to float freely. The hydraulic system FDI sensed the loss of pressure and placed the other actuator on that subsystem (i.e., right inboard flaperon) into damped by-pass. Finally, Impairment No. 20 was a hydraulic pump failure causing all four simplex actuators on System No. 2 to be placed into damped by-pass.

| SURFACE → | | | | | | |
|---|---|---|---|---|---|---|
| IMPAIRMENT NO. | | | | 3 | 1 | 1 3 5 |
| IMPAIRMENT: | 1 | 4 | 9 | 4 | 7 | 7 |
| • 50% BATTLE DAMAGE | 1 | 6 | 11 | 16 | N/A | N/A |
| • 100% BATTLE DAMAGE | 2 | 7 | 12 | 17 | N/A | N/A |
| • SURFACE LOCKED AT TRIM | 3 | 8 | 13 | N/A | N/A | N/A |
| • FLOATING SURFACE | 4 | 9 | 14 | N/A | N/A | N/A |
| • RUNAWAY ACTUATOR | 5 | 10 | 15 | N/A | N/A | N/A |
| • HYDRAULIC SYS FAILURE | N/A | N/A | N/A | 18 | 19 | 20 |

RM89-013

**Fig. 12 Aircraft Impairments Tested**

## 3 - RESULTS

### 3.1 COOPER-HARPER RATINGS

Cooper-Harper ratings assigned to the nominal CRCA are summarized in Table 1. From this point, impairments were injected, ratings determined, and the process repeated with the RS enabled. The incremental improvement in Cooper-Harper ratings assigned at the STOL flight condition are summarized in Fig. 13.

Most improvement in aircraft handling was seen in cases of locked and runaway canard actuator failures and instances of substantial wing trailing edge damage. Substantial wing trailing edge damage was defined as FDIE determination of loss of effectiveness of more than 33%. These determinations occurred in three cases where two flaperons were involved:

- 50% loss of effectiveness of two adjacent flaperons (33%)
- Complete loss of two adjacent flaperons (66% loss of wing TE devices)

**TABLE 1 BASELINE ROBUST CONTROL LAW/AIRCRAFT COOPER-HARPER RATINGS**

| | FLIGHT CONDITION | | |
|---|---|---|---|
| PILOT NO. | ACM ENTRY | TF/ TA | STOL |
| 1 | 1 | 1 | 2 |
| 2 | 1 | 1 | 1 |
| 3 | 4 | 4 | 4 |
| 4 | 5 | 5 | 7 |

RM89-0749-024

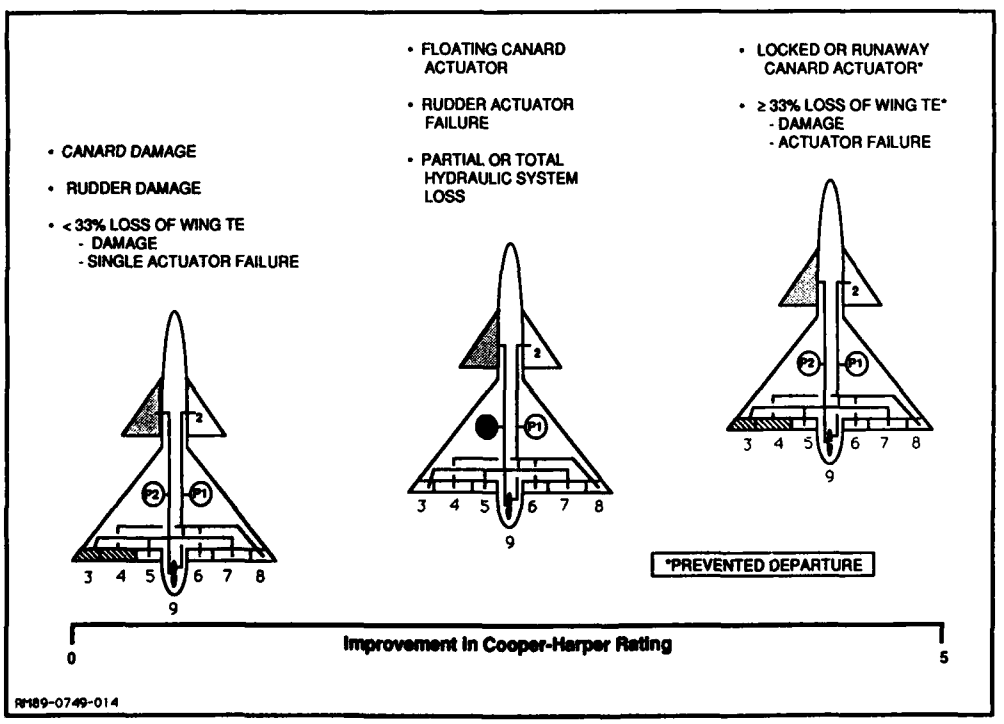



**Fig. 13 STOL Flight Condition Results**

- Ballistic damage causing complete loss of the inboard flaperon and loss of hydraulic pressure to the outboard flaperon (66%).

The C-H ratings typically improved on the order of two to four.

Pilot rating improvements with RTR were somewhat less dramatic in cases of rudder actuator failures and hydraulic system failures; typical rating improvements ranged from one to three for these impairments.

Little or no improvement, or in some cases a decrease in pilot ratings, were observed when activating RTR in cases of canard damage, rudder damage, and loss of wing TE device effectiveness of 33% or less. This last impairment occurred in one of two ways: 1) a flaperon actuator failure; or 2) ballistic damage causing 100% loss of effectiveness of the inboard flaperon.

It should be noted that offline tests peformed prior to LAMARS entry demonstrated instances of FDIE anomalies for a variety of rudder impairments at all flight conditions. These anomalies were attributed to difficulty in differentiating between rudder and canard impairment signatures. However, due to schedule constraints, final tuning of the rudder FDIE was forgone and piloted results were noted accordingly.

Figure 14 illustrates the effects the RS had for impairments tested at the terrain following/terrain avoidance (TF/TA) flight condition. Total hydraulic system loss was seen to have the most improved handling qualities with the advent of RTR at this flight condition. In fact, without RTR, only one of the four pilots was able to avoid departure.

The increments in pilot ratings were somewhat smaller in cases of canard damage, most canard actuator failures, partial hydraulic system loss, and rudder actuator failures. However, even though the increments were smaller, they sometimes included departure prevention.

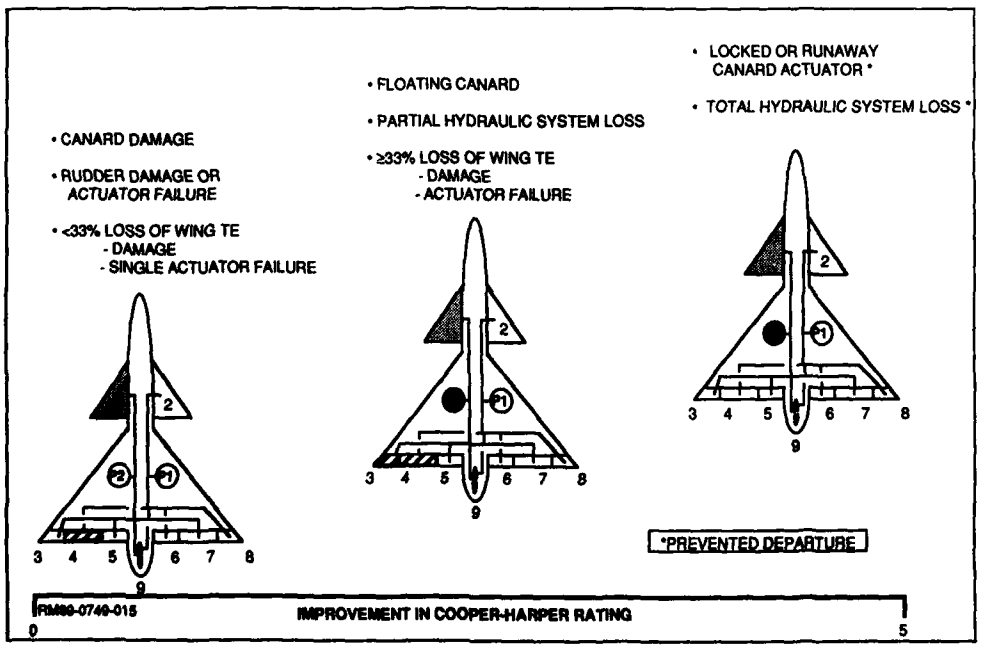



**Fig. 14 ACM Entry Flight Condition Results**

Finally, little or no improvement was seen in the wing trailing edge device impairment scenarios and cases of rudder damage. This was due to the high dynamic pressure and low angle-of-attack characterizing this flight condition. Because of the high dynamic pressure, surplus aileron power existed. Conversely, the low angle-of-attack reduced the need for yaw power to coordinate rolls; therefore loss of yaw power was more difficult to perceive.

Figure 15 graphically presents the improvement in handling qualities seen when invoking reconfiguration at the ACM Entry flight condition. ACM Entry proved to be the most forgiving of the four flight conditions in terms of CRCA handling qualities when impaired. The surplus control power and favorable trim conditions (i.e., dynamic pressure and altitude) translated to basic configuration/control law robustness. Thus, for many impairments the degradation in handling qualities was not as dramatic as other flight conditions and the improvement with RTR was not as significant.

Real-time reconfiguration showed the most improvement in handling qualities for runaway and locked canard actuator failures and in cases of total hydraulic system failures. The incremental improvement in Cooper-Harper ratings ranged from two to five. Reconfiguration limited differential canard deflection and provided departure prevention for canard actuator failures. Damping and controllability were restored in cases of hydraulic system failures.

Somewhat less performance improvement was seen when reconfiguration was employed in cases of floating canard actuator failures, partial hydraulic system failure, and loss of wing trailing edge effectiveness of more than 33%. RTR reduced the cross-coupling seen in cases of a floating canard or partial hydraulic system loss, and restored roll control in instances of substantial flaperon damage.

Little or no change in handling qualities was noted when reconfiguration was added to cases of canard damage, rudder impairments, or wing trailing edge loss of effectiveness equal to 33% or less. The dynamic pressure at this flight condition is

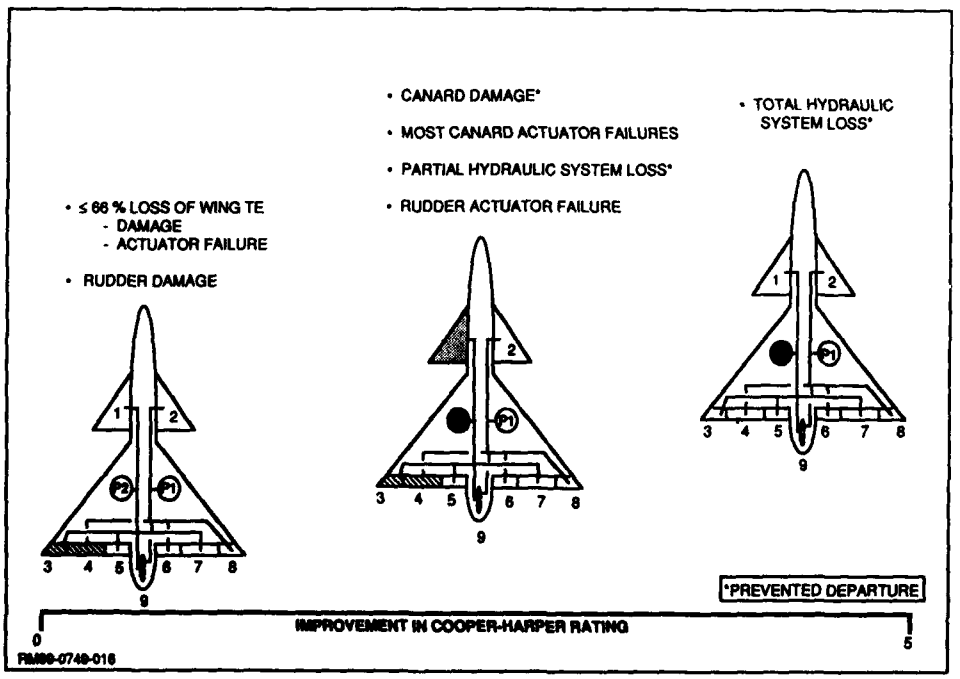



Fig. 15 TF/TA Flight Condition Results

much less than that for TF/TA; therefore, the lateral/directional effects of canard damage are much less pronounced. In turn, reconfiguration's effects are less noticeable. Rudder FDIE anomalies often caused inappropriate reconfiguration of control laws, preventing RTR from aiding the pilot. Loss of one flaperon's effectiveness did not greatly degrade handling qualities and, therefore, left little room for noticeable improvement when reconfiguring.

### 3.2 PILOT WORKLOAD MEASUREMENT

Two flight condition tasks provided measurable parameters in addition to Cooper-Harper ratings. They were final approach to an altitude of 100 ft over the runway threshold at the STOL flight condition, and target tracking at ACM entry. The TF/TA task was not sufficiently defined to ensure repeatability. A "roadway in the sky" display on the HUD may have provided the necessary repeatability, but it would have required additional mechanization.

In a previous Grumman study for the U.S. Navy (Ref 13), a method of measuring pilot longitudinal stick activity was developed to determine workload reduction provided by an automatic carrier landing system (ACLS). This method was modified slightly and used to assess the effects of reconfiguration in all three axes of control.

During the STOL approach-to-landing task, pilot control activity was measured by monitoring sidestick and rudder pedal movement. Longitudinal and lateral sidestick and rudder pedal forces were passed through washout filters, all with 0.5-sec time constants. Thus, each time the pilot moved a controller in the cockpit, it registered as a spike when plotted as a function of time (Fig. 16). The washout removed any steady-state forces the pilot may have been holding since these correspond to a trim position (which could be held automatically if a trim switch existed).

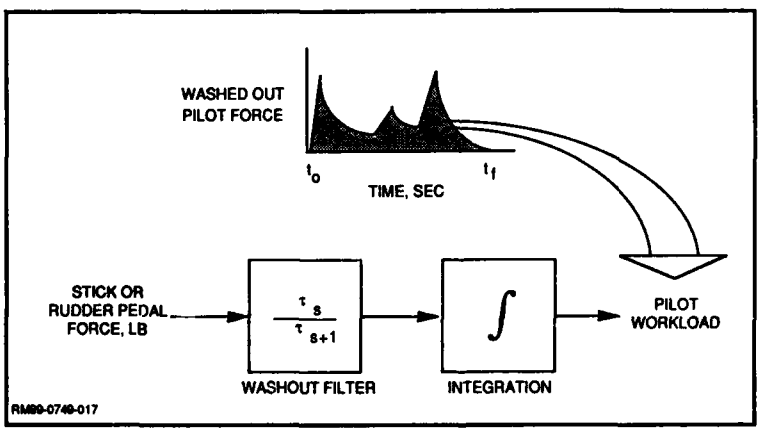



**Fig. 16 Pilot Workload Computation**

Filter outputs were integrated over the period of the tracking task, providing a value for the area under the curve. These areas then became that pilot's total workload in each axis for the given task. The workload numbers have no significance in themselves for the baseline configuration, but they do provide a relative measure when comparing the effects of various impairment/reconfiguration schemes for the same task.

Relative pilot workload was calculated by referencing measured control activity for a given failure in each aircraft axis to that of the baseline configuration. An overall

value of pilot workload was then determined by calculating a square root of the sum of the squares of these values, as shown in the following equation.

$$\text{Overall Workload} = 100\ \frac{\sqrt{\text{Pitch Work}^2 + \text{Roll Work}^2 + \text{Yaw Work}^2}}{\sqrt{(100\%)^2 + (100\%)^2 + (100\%)^2}}$$

It became necessary to first convert the relative workload parameters into percent of baseline values before combining, to avoid mixing units of measure when calculating an overall value (e.g., stick displacement in inches and rudder pedal work in pounds).

Table 2 shows the relative pilot workload registered by Pilot No. 3 when flying final approach in the STOL configuration. This pilot was initialized at an altitude of 1,200 ft and a distance of approximately 4 miles from the runway. A 1% probability of exceedence turbulence model was activated and impairments injected within 5 sec of release.

**TABLE 2 PILOT NO. 3 RELATIVE WORKLOAD MEASUREMENT FOR IMPAIRMENTS AT STOL**

| CONFIGURATION | PITCH | ROLL | YAW | OVERALL | C-H RATING |
|---|---|---|---|---|---|
| BASELINE | 100 | 100 | 100 | 100 | 4 |
| CASE NO. 2/NO RTR | 121 | 195 | 507 | 322 | 9 |
| CASE NO. 2/RTR | 97 | 119 | 302 | 196 | 7 |
| CASE NO. 4/NO RTR | 80 | 154 | 273 | 186 | 7 |
| CASE NO. 4/RTR | 69 | 86 | 50 | 70 | 5 |
| CASE NO. 19/NO RTR | 67 | 161 | 252 | 177 | 8 |
| CASE NO. 19/RTR | 61 | 130 | 192 | 139 | 6 |
| CASE NO. 20/NO RTR | 89 | 167 | 265 | 188 | 8 |
| CASE NO. 20/RTR | 97 | 166 | 33 | 113 | 5 |

RM89-0749-025

Real-time reconfiguration greatly reduced pilot workload in the event of complete canard loss (Case No. 2). In the non-reconfigured aircraft configuration, the asymmetry introduced by loss of the canard was greatly aggravated by the freedom of the healthy canard's movement. When RTR was invoked, right canard activity was reduced while, at the same time, rudder gain was increased with respect to the baseline. The result was a drop in rudder pedal workload of 200%. Overall workload decreased from triple to double the baseline value, and correlates rather well with the pilot's perception as reflected by the Cooper-Harper ratings.

In the event of a canard actuator failure to float (Case No. 4), the pilot's overall workload increased by 86% over the baseline approach. This was due mainly to the asymmetry in yaw due to the effective differential canard deflection that the failure introduced.

When real-time reconfiguration was enabled, all workload parameters were less than 100% of the baseline values. One might, therefore, expect a Cooper-Harper rating that was better than the baseline rating; however, the pilot's rating was C-H = 5 (vs C-H = 4 for the baseline aircraft). This contradicted the measured values, indicating that the pilot felt some increased difficulty when performing the landing task. Analysis of pilot

comments and aircraft parameters showed that RTR removed his need to actively control aircraft heading (constant yaw rate) and reduced cross-coupling, as indicated by the workload parameters, but resulted in a steady-state sideslip angle. By reducing the pilot's active workload, RTR provided him the luxury of directing his attention to other parameters. In this case, sideslip angle was considered uncomfortable. Therefore, this configuration would be expected to be graded as somewhat less desirable than the baseline.

The overall workload value for Case No. 20 (complete hydraulic system loss) with RTR is only 13% higher than that of the baseline; however, the rating of five indicates a noticeable difference to the pilot. Examination of the workload parameters on a "per axis" basis shows that there was no change in lateral axis control work; hence, the degraded rating with respect to the baseline.

### 3.3 TARGET TRACKING SCORES

Figure 17 illustrates the CRCA impairments tested in the ACM target tracking task. This figure is used as a legend for interpretation of subsequent figures.

The plots of Fig. 18 through 22 follow the analogy of a gun pipper in air-to-air gunnery. The outermost circle corresponds to 0% hits (normalized to baseline score). The next concentric circle symbolizes 50% of baseline score. The innermost circle was added for clarity and indicates a score equal to, or greater than, 90% of nominal (scores greater than 100% were achieved). The desired trend is for RTR to cause the symbols to move toward the center of the pipper (dark symbols indicate reconfigured CRCA).

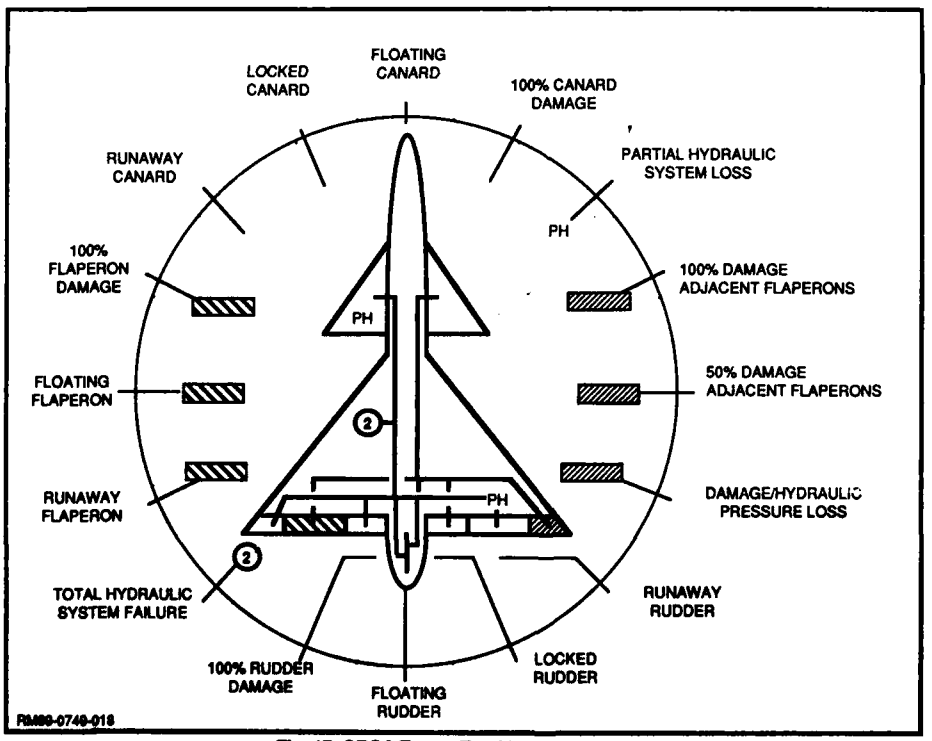



**Fig. 17 CRCA Target Tracking Impairments**

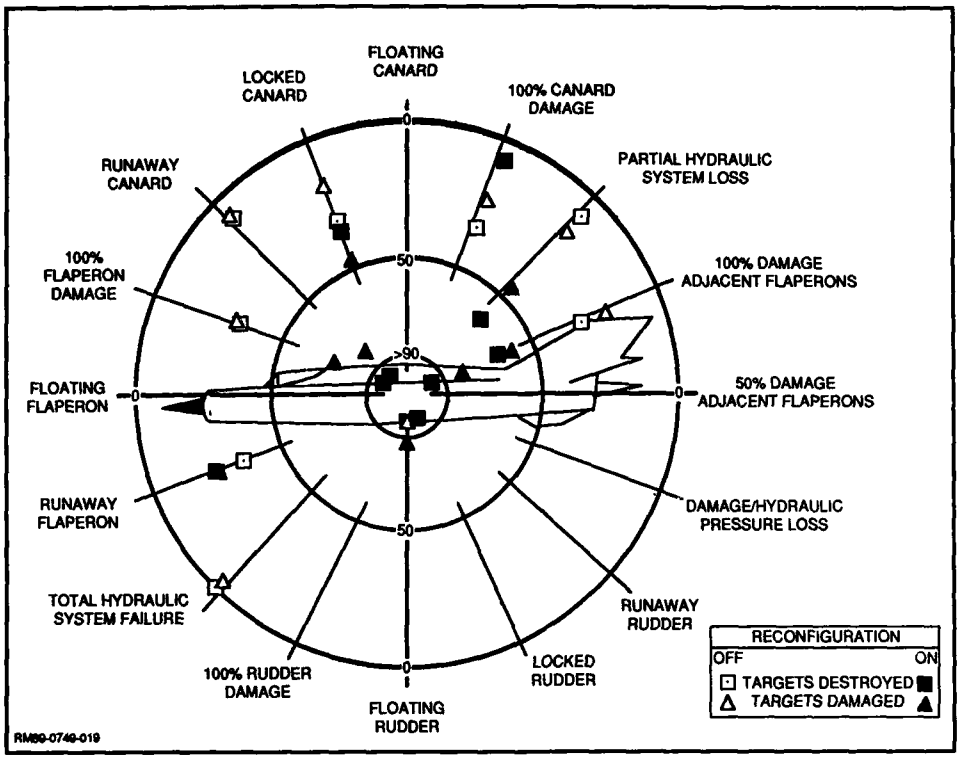



Fig. 18 Pilot No. 1 Performance (% Baseline Score)

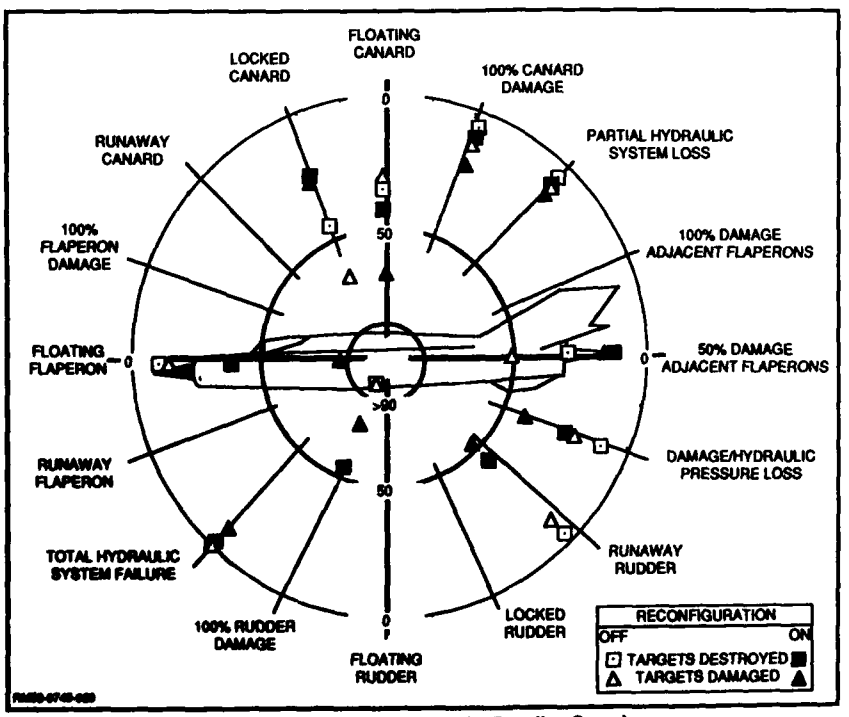



Fig. 19 Pilot No. 2 Performance (% Baseline Score)

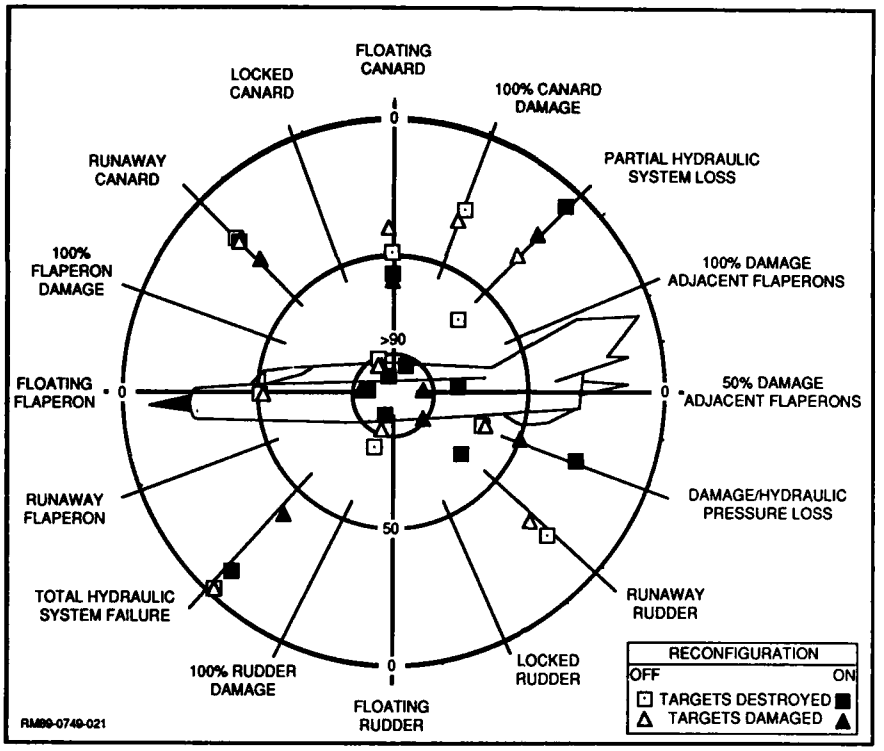



**Fig. 20 Pilot No. 3 Performance (% Baseline Score)**

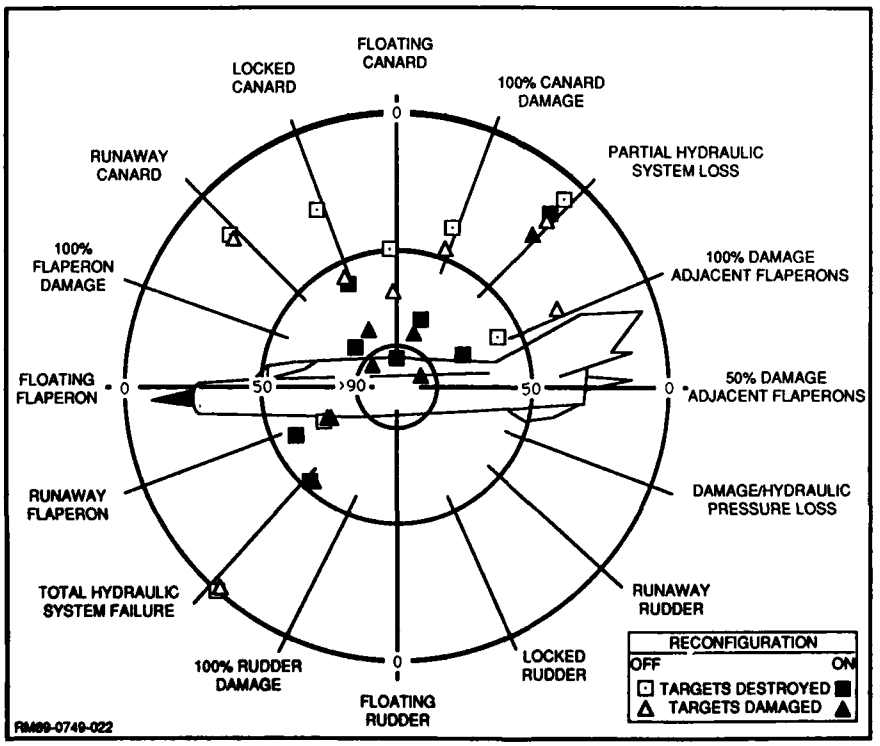



**Fig. 21 Pilot No. 4 Performance (% Baseline Score)**

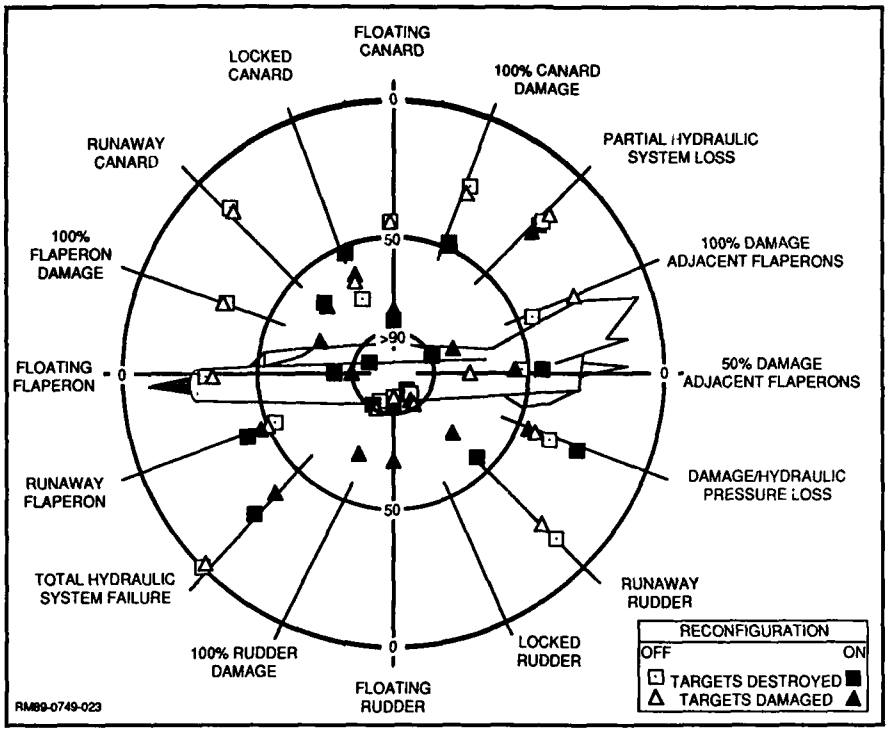



**Fig. 22 Averaged Pilot Performance (% Baseline Score)**

Figures 18 through 21 show the effects of reconfiguration for individual pilots flying the ACM target tracking task under impairments. These scores are referenced to each evaluator's baseline score. When attempting to track with a hydraulic system failure, without reconfiguration capability, three of the four pilots departed controlled flight. After reconfiguration, departure was prevented and scores changed noticeably.

Overall, the general trend in these plots is for migration of the dark symbols toward the center of the pipper. Those instances in which the trend is the opposite of the desired are related to FDIE operation, pilot technique, and six ses were attributed to the reconfigured control laws themselves.

Rudder impairments correctly detected by the FDIE showed improved scores with reconfiguration. Those with no change, or a slight drop, were directly attributed to rudder FDIE anomalies. When impairments were not detected, the differences result from pilot technique.

Figure 22 indicates the average of the four pilots' changes in gunnery hits that damaged (triangles) and those that destroyed the target. The most benefit is seen in cases of: 1) complete hydraulic system failure; 2) loss of canard power via damage or runaway and floating actuator failures; 3) loss of wing flaperon power via a floating actuator or damage to one or both wing flaperons; and 4) runaway rudder actuator failures. Little change in pilot scores was seen for rudder damage and locked or floating rudder actuators. This was due, in part, to missed or delayed detections by the rudder FDIE. However, most of these scores were near or greater than 100% of the baseline scores. This can be attributed to the redundancy of the baseline control law that used canards and rudder for directional control.

### 3.4 CONTROL LAW/FDIE PERFORMANCE SUMMARY

Reconfiguration of the control laws provided most improvement for impairments that caused large reductions in aircraft control response and damping and/or high degrees of cross-coupling of single-axis inputs. Typically, these were cases of hydraulic system failure and canard "hard" actuator failures (i.e., locked or runaway) across the subsonic test envelope. Cases of canard damage showed most improvement at high dynamic pressures because the effects of asymmetries manifested themselves as undesirable side force and lateral accelerations in the cockpit. These effects decreased with decreasing dynamic pressure.

Reconfiguration for loss of wing trailing edge device effectiveness of 33% or more of one wing's contribution showed increasing improvement with decreasing dynamic pressure. This occurred because roll power became more critical and reconfiguration's benefits more noticeable.

Reconfiguration for rudder impairments showed varying degrees of success. This result was a combination of the robustness of the aircraft/control law configuration (e.g., canards and rudder used for directional control at most flight conditions) and FDIE operation.

Slight degradation in handling qualities or target tracking scores (not related to FDIE operation) was noted in 6.7% of the cases tested. Often, the drop in target tracking scores was in fine tracking (i.e., "destroy" hits) with no change in gross acquisition peformance (i.e., "damage" hits). Instances where a drop in Cooper-Harper rating was noted by one pilot (with reconfiguration enabled), were usually contradicted by other pilots' ratings.

The occurrence of FDIE anomalies over the course of the LAMARS simulation are summarized in Table 3. Recall that rudder-related FDIE had not progressed through the final tuning process prior to verification tests. Therefore, rudder FDIE anomalies (indicated in superscripts) were seen to account for a majority of the cases shown here. Eliminating these cases shows that reconfiguration based on FDIE anomalies caused a drop in aircraft performance in 1.1% of the tests. All such instances were related to false isolation of damage.

Clearly, no false alarms occurred throughout the course of the evaluation. By definition, this means that FDIE never reported the existence of a problem when flying the nominal CRCA. Out of all the tests performed, local FDIE missed detecting actuator impairments in 0.6% of the tests and no instance of degraded performance resulted. Here, all missed actuator detections were rudder actuator failures. Ballistic damage was not detected in 1.1% of the cases; all were cases of rudder damage and performance degradation resulted in half of those cases.

**TABLE 3 SUMMARY OF FDIE ANOMALIES**

| | % OF TOTAL CASES TESTED | | | |
|---|---|---|---|---|
| ANOMALY | LOCAL | DROP IN PERFOR-MANCE | GLOBAL | DROP IN PERFOR-MANCE |
| FALSE ALARMS | 0.0 | 0.0 | 0.0 | 0.0 |
| MISS | $0.6^{(0.6)}$ | 0.0 | $1.1^{(1.1)}$ | $0.6^{(0.6)}$ |
| DELAY | $1.1^{(0.6)}$ | 0.0 | 0.0 | 0.0 |
| FALSE ISOLATION | $1.1^{(0.6)}$ | $0.6^{(0.6)}$ | $7.7^{(2.2)}$ | $2.2^{(1.1)}$ |
| MIS-ESTIMATES | -- | -- | $1.1^{(1.1)}$ | $0.6^{(0.6)}$ |
| RM89-0749-028 | ( ) - INDICATES RUDDER-RELATED FDIE PERFORMANCE | | | |

Delays in detecting actuator failures occurred in 1.1% of the total cases tested. Half of these were affiliated with the rudder. False isolations were the most frequent of the FDIE anomalies. Local false isolations (i.e., actuator failure attributed to the wrong surface as either actuator failure or damage) occurred in 1.1% of the cases. Rudder FDIE was the cause for half of these cases and instances of degradation in performance. Global mis-isolations were the most frequently noted, occurring 7.7% of the time, with 2.2% attributed to rudder. Performance dropped as a result of these anomalies in 2.2% of the test cases, with half attributed to the rudder. Mis-estimates refer to damage estimates that differ by more than 33% of actual control surface damage. This anomaly only occurred in cases of rudder damage (1.1%), causing performance to be degraded in half those cases.

### 3.5 POSITIVE PILOT ALERT

Two general trends were noted when the positive pilot alert (PPA) system was activated in conjunction with the robust control laws (RCLs) and FDIE algorithms. PPA's appearance provided useful information to the pilot without impeding the task at hand, and it caused pilot ratings to improve in all but one impairment case. The one exception occurred when the pilot thought handling qualities were nominal, but was notified of damage to wing trailing edge devices. His comment was, "PPA says I have a problem. I can't feel it, but I'm going to give it a 3." Up to this point, the pilot's rating was 1 on the Cooper-Harper scale.

The AAD automatically drew each pilot's attention to the existence of an impairment and a change in aircraft limits. Usually the AAD was immediately cleared from the HUD to reduce clutter. The pilots were then able to select from three display configurations on the multi-function display (MFD). As time and pilot workload allowed, the MFD menus were called up. The most favored was the control system status (CSS) display, which one pilot described as, "telling me 90% of what I need to know about my aircraft." Initial pilot response to the flight status display was that it appeared busy. However, once the evaluators became familiar with it, they felt all information provided was pertinent and could not offer any parameters for deletion. In fact, because real-time reconfiguration provided a somewhat different aircraft for each impairment (resulting in a large matrix of possible limits) it was felt that such a display would be necessary.

The emergency procedures display was unconditionally accepted by all pilots. Here, also, it was felt that such a menu was required when considering the matrix of characteristics reconfiguration could provide. Both the emergency procedures and flight status displays were menus that would be selected in lower workload environments.

## 4 - CONCLUSIONS

Piloted tests conducted at the USAF LAMARS facility verified the capability of a reconfiguration strategy to extend combat persistence in the event of control surface impairments. Reconfigurable control laws worked in concert with failure detection, isolation, and damage estimation algorithms to provide improved control when compared to a set of non-reconfigured robust control laws. The positive pilot alert system increased pilot awareness of impairments and provided the requisite information to permit timely adaptation.

The baseline control system provided robustness to relatively benign impairments, making the effects of reconfiguration difficult to discern. Impairments of increasing signature showed increased benefits from reconfiguration, including departure prevention in some cases. The greatest benefit of control law reconfiguration was seen for hydraulic system failures, canard actuation failures, canard damage at high dynamic

pressure, and wing trailing edge damage at low speeds. Local FDIE was successful in detecting and isolating actuator failures, while the global FDIE provided satisfactory estimates of damage for the more severe cases.

This evaluation reflects the first involvement of pilots in development of the control laws and FDIE. The PPA received high marks and experienced few instances of criticism over the course of the evaluation. Its success was due, in large part, to the fact that Grumman test pilots were involved in its design. Some re-ordering of the menus displayed is warranted; however, these menus were highly successful in providing the pilot with familiarity with the new configuration resulting from reconfiguration. It is felt that iteration of the reconfigurable control laws and FDIE based on this piloted simulation will provide a similar degree of success.

## ACKNOWLEDGEMENTS

This paper presents the results of efforts of a large group of people. A number of associate contractors were involved in the design, development, and testing of the reconfiguration strategy. The author wishes to acknowledge the following key personnel for their contributions: Mr. Robert Quaglieri, Capt. Robert Eslinger, and Mr. Phil Chandler, of WRDC/FIGX, for overall program management and guidance; and Messrs. Charles Boppe, Howard Berman, R. Paul Martorella and Warren Weinstein, of Grumman Aircraft Systems, for technical and managerial guidance and support. In addition, the efforts of those who developed the reconfigurable control laws at Lear Astronics, the FDIE algorithms at Charles River Analytics, and the PPA system at Grumman are also recognized. A great deal of appreciation and thanks goes to the engineers of WRDC/FIGL and Century Computing, Inc. for their tireless efforts in integrating the software into a useful verification tool. Finally, on behalf of all designers, particular thanks is extended to the pilots: Majors Tom Gilkey, Dave Carlson, John Voss, and George Wissler (CAF), from USAF HQ TAC, Major Allen Reed of the USAF TPS faculty, and Mr. Bill Dana of NASA/Dryden for the time and expertise they provided.

## REFERENCES

1) Berman, H.L., and Boudreau, J.A., "Dispersed And Reconfigurable Digital Flight Control System," AFFDL-TR-79-3125, December 1979.

2) "Self-Repairing Digital Flight Control System Study, Final Report - Part 1," General Electric Co., AFWAL-TR-88-3007, May 1988.

3) "Reconfiguration Strategies For Aircraft Flight Control Systems Subjected To Actuator Failure/Surface Damage," Scientific Systems, Inc., AFWAL-TR-86-3079, December 1986.

4) Weinstein, W., et al., "Control Reconfigurable Combat Aircraft (CRCA) Development, Phase I - R&D Design Evaluation," AFWAL-TR-87-3011, May 1987.

5) Weinstein, W., and Mercadante, R., "Control Reconfigurable Combat Aircraft, Volume I: Architecture and Simulation Development," AFWAL-TR-88-3118, July 1989.

6) "Reconfigurable Control Laws for the Control Reconfigurable Combat Aircraft Subjected to Actuator Failures and Surface Damage," Lear Astronics, Inc., WRDC-TR-89-3052, March 1989.

7) Caglayan, A.K., et al., "Failure Detection, Isolation and Estimation for the Control Reconfigurable Combat Aircraft Subjected to Actuator Failures and Surface Damage," Charles River Analytics, Inc., WRDC-TR-89-3058, March 1989.

8) Mercadante, R., "Control Reconfigurable Combat Aircraft, Volume II: Piloted Test Results," Grumman Aircraft Systems Division, WRDC-TR-89-3081, June 1989.

9) Maybeck, P.S., "Failure Detection through Functional Redundancy," AFFDL-TR-76-93, September 1976.

10) Caglayan, A.K., et al., "A Hierarchichal Reconfiguration Strategy For Aircraft Subjected To Actuator Failure/Surface Damage," AFWAL-TR-87-3024, May 1987.

11) "MIL-F-8785C, Military Specification - Flying Qualities of Piloted Airplanes," February 1980.

12) Cooper, G. E., and Harper, R.P., Jr., "The Use of Pilot Rating in the Evaluation of Aircraft Handling Qualities, "NASA TN D-5153, April 1969.

13) Martorella, R.P., "Precision Flight Path Control Laws, Final Report For Contract N00421-83-C-0069," 15 June 1983.

# FLIGHT TEST RESULTS OF FAILURE DETECTION AND ISOLATION ALGORITHMS FOR A REDUNDANT STRAPDOWN INERTIAL MEASUREMENT UNIT

by

**F.R.Morrell**
NASA Langley Research Center
Mail Stop 489
Hampton, VA 23665-5225
United States

**P.R.Motyka**
The Charles Stark Draper Laboratory, Inc.
Cambridge, MA 02139
United States

**M.L.Bailey**
PRC Kentron International
Mail Stop 489
Hampton, VA 23665-5225
United States

## ABSTRACT

Flight test results for two sensor fault-tolerant algorithms developed for a redundant strapdown inertial measurement unit are presented. The IMU consists of four two-degrees-of-freedom gyros and accelerometers mounted on the faces of a semi-octahedron. Fault tolerance is provided by edge vector test and generalized likelihood test algorithms, each of which can provide dual fail-operational capability for the IMU. To detect the wide range of failure magnitudes in inertial sensors, which provide flight crucial information for flight control and navigation, failure detection and isolation are developed in terms of a multi level structure. Threshold compensation techniques, developed to enhance the sensitivity of the failure detection process to navigation level failures, are presented.

Four flight tests were conducted in a commercial transport-type environment to compare and determine the performance of the failure detection and isolation methods. Dual flight processors enabled concurrent tests for the algorithms. Failure signals such as hard-over, null, or bias shift, were added to the sensor outputs as simple or multiple failures during the flights. Both algorithms provided timely detection and isolation of flight control level failures. The generalized likelihood test algorithm provided more timely detection of low-level sensor failures, but it produced one false isolation. Both algorithms demonstrated the capability to provide dual fail-operational performance for the skewed array of inertial sensors.

## NOMENCLATURE

| | |
|---|---|
| $a_i^B$ | resolution of accelerometer output in body axes, $B = x, y, z;\ i = 1, 2, 3, 4$; rad/sec$^2$ |
| $a_x, a_y, a_z$ | lateral, longitudinal, and normal body axes accelerations, ft/sec$^2$ |
| $DF_D$ | GLT failure decision function, rad$^2$ or (ft/sec)$^2$ |
| $DF_{I_j}$ | GLT failure isolation function for the jth sensor, rad$^2$ or (ft/sec)$^2$ |
| $Dt$ | computation time interval, sec |
| $e_{ij}$ | edge vector relating $i$th and $j$th instruments, $j > i;\ i = 1, 2, 3$ |
| $g$ | acceleration due to gravity, ft/sec$^2$ |
| $H$ | sensor configuration geometry matrix |
| $m$ | sensor output, rad or ft/sec |
| $p_{a_i}$ | vector of GLT accelerometer parity residuals, ft/sec; $i = 1, \ldots, n-3$ |
| $p_{a_{ij}}$ | edge vector accelerometer parity residual, ft/sec; $j > i; i = 1, 2, 3$ |
| $s_{a_i}$ | accelerometer measurement output, ft/sec; $i = x1, y1, \ldots x4, y4$ |
| $s_i$ | spin/pendulous axes of inertial sensor $i$ |
| $t$ | time, sec |
| $T$ | failure detection threshold, rad or rad$^2$, ft/sec or (ft/sec)$^2$ |
| $V$ | (n−3) x n matrix of parity coefficients |
| $x_i, y_i$ | sensor input axes, $i = 1, 2, 3, 4$ |
| $\alpha, \beta, \gamma$ | direction cosines |
| $\epsilon$ | scale factor error, ppm |
| $\lambda$ | bias error, ft/sec$^2$ |
| $\mu$ | misalignment error, rad |
| Subscripts | |
| $a$ | accelerometer |
| $f$ | filtered valued |
| $g$ | gyro |
| $h$ | flight control or hard-level |
| $m$ | positive maximum or upper bound |
| $s$ | navigation or soft-level |
| $w$ | washout filter |
| $x, y, z$ | body axes components |

Superscripts

| | |
|---|---|
| $T$ | transpose |

## 1. INTRODUCTION

Integrated avionics concepts for advanced aircraft may use strapdown inertial sensors for angular rate and linear acceleration measurements for flight control and navigation systems. To meet safety and reliability requirements, triply redundant IMU's are typically used in the design of operational units. Future aircraft designs, however, will stress maximum efficiency and relaxed static stability requiring flight crucial data from the inertial sensors for integrated avionics functions. Reliability and safety issues for these aircraft will mandate automatic selection of operational sensors and quick rejection of failed components so that flight control and navigation performance will not be impaired. Accordingly, research at NASA Langley Research Center has been directed toward fault-tolerant concepts for inertial sensor arrays. Cost, weight, and power considerations may restrict the system complexity and number of inertial sensors available to generate flight critical data. This implies the use of an optimum geometric array of sensors with efficient fault-tolerant algorithms to satisfy safety and reliability requirements (ref. 1).

Fault tolerance can be provided through the implementation of failure detection and isolation (FDI) algorithms. Two FDI methods have been developed for a redundant strapdown inertial measurement unit (RSDIMU), built for NASA Langley Research Center. This unit consists of a semi-octahedral array of four two degrees-of-freedom (TDOF) gyros and accelerometers with appropriate processing electronics (ref. 2). The FDI algorithms are: 1) the edge vector test (EVT) which is the pairwise comparison of TDOF sensor measurements; and 2) the generalized likelihood test (GLT) where the detection and isolation of failures falls within the framework of composite hypothesis tests (refs. 3–5). Both algorithms are capable of detecting and correctly isolating multiple accelerometer or gyro failures. To account for a wide range of sensor uncertainties and anomalies, a unified multi-level structure providing failure detection capability for flight control level failures as well as navigation level failures of accelerometers or gyros is implemented (ref. 6). Each FDI algorithm is designed to provide fail-operational/fail-operational/fail-safe capability for the RSDIMU, timely detection and isolation of sensor failures, reduced false alarm rate, and real-time operation in flight computers.

The goal of the present FDI development has been to provide detection and isolation of inertial sensor failures before flight crucial control system data are corrupted. For flight control purposes failure detection merely requires that a function of the sensor outputs be compared to a constant failure threshold. A second goal has been to develop FDI for low-level failures which can be tolerated for flight control purposes but are unacceptable for navigation. For lower-level failures, however, the failure detection process is more complicated. The problem is that maneuvering flight excites sensor uncertainties (e.g., uncompensated scale factor and misalignment) to a greater extent than in cruise. To avoid false detections of low-level failures during maneuvers, therefore, some form of failure threshold compensation is required. This paper will discuss the development of failure threshold compensation for both the EVT and the GLT algorithms.

The topics covered in this paper include the flight demonstration and evaluation of the two FDI algorithms developed for the RSDIMU. The development of the parity equations, failure thresholds, and isolation functions for accelerometer FDI is presented. A brief description of the RSDIMU, the software processing scheme for the FDI algorithms, and the flight test hardware is given. The performance of the FDI algorithms in flight for a variety of injected sensor failures, which would affect flight control and navigation functions, completes the paper.

## 2. DESCRIPTION OF THE RSDIMU

The RSDIMU shown in figure 1 has a complement of four TDOF gyros and accelerometers mounted in a semi-octahedral configuration such that dual fail-operational performance is realizable. The spin/pendulous axes, $s_i$, of the sensors (fig. 2) are normal to the faces of the semi-octahedron, while the measurement axes, $x_i, y_i, (i = 1, 2, 3, 4)$, lie in the planes of the faces and are oriented such that the bisector of the sensitive axes is normal to the baseline of the semi-octahedron. The RSDIMU consists of two separable but communicating packages (faces 1 and 2, faces 3 and 4) which may be spatially separated along a track in the lateral direction for damage control. For the flight tests considered in this paper, however, the two units were collocated as shown in figure 1. The outputs of any two gyros/accelerometers constitute sufficient information to complete an orthogonal triad of angular rate/linear acceleration body frame solutions. From figure 2, the ideal coordinate transformation which relates the sensor measurement axes to the body frame axes $(x, y, z)$ is

$$H = \begin{bmatrix} -\alpha & \beta & -\gamma \\ \beta & -\alpha & -\gamma \\ \beta & \alpha & -\gamma \\ -\alpha & -\beta & -\gamma \\ \text{---} & \text{---} & \text{---} \\ \alpha & -\beta & -\gamma \\ -\beta & \alpha & -\gamma \\ -\beta & -\alpha & -\gamma \\ \alpha & \beta & -\gamma \end{bmatrix} \tag{1}$$

where $\alpha = \left(\sqrt{3}+1\right)/2\sqrt{3}$, $\beta = \left(\sqrt{3}-1\right)/2\sqrt{3}$, *and* $\gamma = 1/\sqrt{3}$. The dashed line of the matrix $H$ indicates the separation of the IMU into halves: IMUA (instruments 1 and 2) and IMUB (instruments 3 and 4). In the ideal case the accelerometer

and gyro direction cosines are the same. The 4 × 3 matrix $H_s$ defines the ideal transformation from the gyro/accelerometer spin/pendulous axes $(s_1, \ldots, s_4)$ to the body frame $(x, y, z)$ as

$$H_s = \begin{bmatrix} -\gamma & -\gamma & \gamma \\ -\gamma & \gamma & \gamma \\ \gamma & \gamma & \gamma \\ \gamma & -\gamma & \gamma \end{bmatrix} \tag{2}$$

The RSDIMU functional block diagram for the flight test configuration is shown in figure 3. Each TDOF accelerometer/gyro pair on a face of the semi-octahedron has an independent microprocessor which processes raw sensor data and provides compensation for sensor errors. The microprocessors transfer time homogeneous data to their respective flight computers (ref. 7). The redundancy management algorithm is processed at the 64 Hz system rate to ensure that valid data are used in the RSDIMU functional outputs. For the flight tests discussed in this paper the EVT algorithm was processed in IMUA and the GLT algorithm was processed in IMUB. A least-squares solution for linear acceleration can be found from the expression

$$\hat{a} = (H^T H)^{-1} H^T m_a / Dt \tag{3}$$

where $H$ is the 4 × 3 matrix formed from the two accelerometers used for the solution and $m_a$ is the measurement sensor vector formed from the two accelerometers.

## 3. FAILURE DETECTION AND ISOLATION CONCEPTS

The purpose of the experimental RSDIMU is to provide a system for evaluating FDI algorithms which provide the dual fail-operational capability necessary to satisfy reliability and performance requirements for flight control and navigation systems (ref. 8). Dual fail-operational performance for the RSDIMU implies the ability to survive two gyro/accelerometer failures and to cease operation when a third failure of either type of sensor is detected. To detect sensor failures a system of parity equations is solved. Parity equations are linear combinations of the sensor outputs selected to enhance uncertainties (uncompensated errors and failures) associated with the sensors. The effects of the quantity the sensors measure (angular rate/linear acceleration) are removed from consideration by the parity equations.

Failure detection occurs as a result of comparing the parity equation residuals or a function of them to a threshold. If the threshold is exceeded, a failure is declared and the failed sensor is then isolated. Several methods are employed to accomplish failure isolation depending on the algorithm employed. Logical operations on the residuals which exceed the threshold isolate the failure to a particular sensor, or the dot product of the vector of parity equation residuals with vectors defined by the parity equation coefficients can be used to isolate a failed sensor.

For flight control level failures the failure detection process is simple. The parity equation residuals are merely compared to a constant level threshold. For the avionics functions which require greater accuracy and thus require the detection of lower level failures, however, the process is more complicated. Figure 4 illustrates the flow of information applicable to low-level FDI processes. The parity equations are formed from current sensor data, filtered, and compared to a threshold. The threshold is formed from several contributions: 1) a constant to account for unfiltered noise and quantization levels, 2) sensor errors which can be computed analytically using statistical data (supplied by the manufacturer), such as uncompensated bias, scale factor, and misalignment errors, and 3) compensation for high frequency residuals. Immediate past values of sensor data are used to form the analytical part of the threshold function to prevent corruption that may be caused by hard failed sensors.

Since the FDI algorithm has been designed to cover sensor failures which affect flight control to navigation levels, a baseline configuration has been suggested as shown in figure 5 (ref. 6). Unfiltered parity equations residuals are processed at the sensor measurement rate (64 Hz) and compared to a constant threshold to ensure the removal of hard-failed sensor data before vehicle controllability is affected. The same parity equation residuals are filtered to attenuate quantization and noise so that failure levels which might affect display (mid-level) or navigation (soft-level) performance might be detected when compared to a compensated threshold. The filtered parity equations are processed at lower rates since these failure levels may be tolerated for a longer period of time without affecting vehicle controllability. The mid-level channel was not implemented for the flight tests.

## 4. INERTIAL SENSOR ERROR MODELS

The sensors used in the RSDIMU are TDOF gyros and accelerometers (ref. 2). The first-order uncompensated errors for the accelerometers used for the flight demonstration include include constant bias, misalignment, and scale factor nonlinearities. An expression for the output of the $x$-axis of accelerometer 1 (not including the effects of quantization and noise) is

$$\begin{aligned} s_{a_{x1}} = [&-\alpha \cdot a_x + \beta \cdot a_y - \gamma \cdot a_z + \lambda_{a_1} - \mu_{a_{11}} \cdot \alpha \cdot a_x + \mu_{a_{12}} \cdot \beta \cdot a_y - \mu_{a_{13}} \cdot \gamma \cdot a_z \\ &+ \epsilon_{a_1} \cdot (-\alpha \cdot a_x + \beta \cdot a_y - \gamma \cdot a_z)] \cdot Dt \end{aligned} \tag{4}$$

where $Dt$ accounts for the accelerometer output (ft/sec/cycle). The first three terms of Eq.(4) represent the output of an ideal instrument, and the remaining terms represent uncompensated first order error. The expression for gyro output with first order errors would be similarly developed.

## 5. FDI ALGORITHM DEVELOPMENT

The techniques used to detect and isolate the presence of failures include the generalized likelihood test and the edge vector test methods. The methodology for the failure detection process is given in figure 4. The FDI schemes are developed for accelerometers; however, the gyro development would be similar.

### 5.1 Edge Vector Test Method

A technique to determine parity equations which are particularly suited to TDOF instruments has been developed (refs. 1, 9). A set of six parity equations for the gyros or accelerometers of the RSDIMU is formed when pairs of instruments are compared along an edge of the semi-octahedron. An expression for the edge vectors is formed from the cross products of the sensor pendulous axes as illustrated in figure 2 is

$$e_{ij} = (s_i \times s_j)/(|s_i \times s_j|) \qquad j > i; \quad i = 1,2,3 \tag{5}$$

The accelerometer parity equations for the vector-based method are defined as the dot product

$$p_{a_{ij}} = \left[\left(a_i^B - a_j^B\right) \cdot e_{ij}\right] \cdot Dt \qquad j > i; \quad i = 1,2,3 \tag{6}$$

To express the quantity $a_i^B$ in terms of sensor output, use Eq.(1) to obtain

$$a_i^B = \left[H^T \cdot s_{a_i}\right] / Dt \tag{7}$$

where $H^T$ is the $3 \times 2$ matrix corresponding to the $i$th accelerometer. This expression can be used to obtain the accelerometer parity equations in terms of sensor measurements. To obtain the parity residuals in terms of sensor errors, substitute Eq.(4) for the measurements. The results are

$$\begin{aligned}
p_{a_{12}} &= 0.966\left(\delta s_{a_{x1}} - \delta s_{a_{y2}}\right) + 0.259\left(\delta s_{a_{y2}} - \delta s_{a_{x2}}\right) \\
p_{a_{13}} &= 0.707\left(\delta s_{a_{x1}} + \delta s_{a_{x3}} - \delta s_{a_{y1}} - \delta s_{a_{y3}}\right) \\
p_{a_{14}} &= 0.966\left(\delta s_{a_{x4}} - \delta s_{a_{y1}}\right) + 0.259\left(\delta s_{a_{y4}} - \delta s_{a_{x1}}\right) \\
p_{a_{23}} &= 0.966\left(\delta s_{a_{x2}} - \delta s_{a_{y3}}\right) + 0.259\left(\delta s_{a_{y2}} - \delta s_{a_{x3}}\right) \\
p_{a_{24}} &= 0.707\left(\delta s_{a_{x2}} + \delta s_{a_{x4}} - \delta s_{a_{y2}} - \delta s_{a_{y4}}\right) \\
p_{a_{34}} &= 0.966\left(\delta s_{a_{x3}} - \delta s_{a_{y4}}\right) + 0.259\left(\delta s_{a_{y3}} - \delta s_{a_{x4}}\right)
\end{aligned} \tag{8}$$

where the $\delta s_{a_i}$ $(i = x1, y1 \ldots x4, y4)$ terms represent total accelerometer error.

The development of gyro parity equations and the reduction to sensor errors is similar to the accelerometer development.

Sensor failure detection is determined from a comparison of the parity equation residuals, or a function of them to a threshold. If the threshold is exceeded, a failure is declared. For the hard-range failures the threshold may be merely a constant. For the mid- or soft-range levels, however, normal dynamic sensor errors, which are more pronounced during maneuvers, are larger than acceptable bias errors. For example, for a standard rate turn (3 deg/sec) a gyro scale factor error of 100 ppm yields an equivalent drift rate of 1 deg/hr. A constant bias level this high would result in serious navigation error.

As suggested by Eq.(4), the accelerometer outputs contain static (bias) and dynamic (misalignment and scale factor) errors. Since these errors are statistically known (e.g. from manufacturers' data), an estimate for the maximum first order accelerometer errors can be written from Eq.(4) as

$$\delta s_{a_m} = \left[\lambda_{a_m} + (\mu_{a_m} + \epsilon_{a_m}) \cdot \left(\alpha \cdot |\hat{a}_{x_f}| + \alpha \cdot |\hat{a}_{y_f}| + \gamma \cdot |\hat{a}_{z_f}|\right)\right] \cdot Dt \tag{9}$$

where the $\hat{a}_{i_f}$ terms are least-squares estimates for acceleration from an appropriate accelerometer pair; these terms are filtered to be compatible with the channel under consideration. An estimate of the maximum accelerometer sensor error contribution to a parity equation residual is obtained from the coefficients of the sensor error terms in Eq.(9) as

$$p_{a_m} = 2.828 \cdot \delta s_{a_m} \tag{10}$$

The constant in Eq.(10) is the sum of the absolute values of the coefficients of the error terms in parity equations $p_{a_{13}}$ and $p_{a_{24}}$ to account for worst case conditions. The sum of the absolute values for the remaining parity equations is 2.445.

Compensation for high frequency terms in the parity residuals caused by unattenuated high frequency terms (noise spikes, etc.) can be obtained by passing each residual through a washout filter, and adding the result to the threshold as indicated in figure 4. This high-pass filter has the effect of driving the low frequency terms to zero while compensating the threshold for high-frequency effects appearing in the residuals; also the filter tends to render the failure detection process false-alarm free.

The total threshold for each accelerometer parity equation consists of a constant term, $T_{aq}$, accounting for quantization and noise effects, a maximum accelerometer error term found from Eq.(9), and a high frequency compensation term, $T_{aw_{ij}}$,

obtained by passing the parity residual through a washout filter. The expression for the threshold for each parity equation is,

$$\begin{aligned} T_{a_{12}} &= T_{aq} + 2.445 \cdot \delta s_{a_m} + T_{aw_{12}} \\ T_{a_{13}} &= T_{aq} + 2.828 \cdot \delta s_{a_m} + T_{aw_{13}} \\ T_{a_{14}} &= T_{aq} + 2.445 \cdot \delta s_{a_m} + T_{aw_{14}} \\ T_{a_{23}} &= T_{aq} + 2.445 \cdot \delta s_{a_m} + T_{aw_{23}} \\ T_{a_{24}} &= T_{aq} + 2.828 \cdot \delta s_{a_m} + T_{aw_{24}} \\ T_{a_{34}} &= T_{aq} + 2.445 \cdot \delta s_{a_m} + T_{aw_{34}} \end{aligned} \tag{11}$$

Each parity equation residual is tested against its corresponding threshold, so that if $(|p_{a_{ij}}| - T_{a_{ij}}) > 0$, the parity equation indicates a failure. A logical flag, $L_{ij}$, is set true for this event. At the completion of all six threshold tests for the parity equations, a set of logical equations is examined to isolate a failed instrument. The logical equations are

$$\begin{aligned} A_1 &= L_{12} \cdot L_{13} + L_{12} \cdot L_{14} + L_{13} \cdot L_{14} \\ A_2 &= L_{12} \cdot L_{23} + L_{12} \cdot L_{24} + L_{23} \cdot L_{24} \\ A_3 &= L_{13} \cdot L_{23} + L_{13} \cdot L_{34} + L_{23} \cdot L_{34} \\ A_4 &= L_{14} \cdot L_{24} + L_{14} \cdot L_{34} + L_{24} \cdot L_{34} \end{aligned} \tag{12}$$

where $\cdot$ denotes the logical "and" function and + denotes the logical "or" function. If sensor 1 fails, the three flags associated with that sensor, (e.g., $L_{12}$, $L_{13}$, and $L_{14}$) will be set true when their respective parity tests are greater than zero. When any two of these flags are set true, $A_1$ will be set true thus isolating accelerometer 1. A redundancy management algorithm removes an isolated sensor failure from further output consideration. Table 1 contains the logic equations used to isolate a second failure. A third failure may be detected but not isolated.

Table 1. Logic equations for isolating a second sensor failure

| Instrument 1 Failed | Instrument 2 Failed | Instrument 3 Failed | Instrument 4 Failed |
|---|---|---|---|
| $A_2 = L_{23} \cdot L_{24}$ | $A_1 = L_{13} \cdot L_{14}$ | $A_1 = L_{12} \cdot L_{14}$ | $A_1 = L_{12} \cdot L_{13}$ |
| $A_3 = L_{23} \cdot L_{34}$ | $A_3 = L_{13} \cdot L_{34}$ | $A_2 = L_{12} \cdot L_{24}$ | $A_2 = L_{12} \cdot L_{23}$ |
| $A_4 = L_{24} \cdot L_{34}$ | $A_4 = L_{14} \cdot L_{34}$ | $A_4 = L_{14} \cdot L_{24}$ | $A_3 = L_{13} \cdot L_{23}$ |

The logic equations presented are applicable to multiple nonconcurrent failures for the gyros or accelerometers. Modifications to the logic to account for multiple concurrent failures can easily be implemented.

5.2 Generalized Likelihood Test Method

The GLT algorithm described in this section follows the development given in ref. 8. The detection of failures is accomplished by processing the sensor measurements in a set of parity equations which have the form

$$p_{a_i} = \sum_j v_{ij} s_{a_j} \qquad i = 1, \ldots n-3; \quad j = x1, y1, \ldots x4, y4 \tag{13}$$

where the $V$ matrix is chosen to be of dimension (n−3) x n with $VH = 0$, and $VV^T = I$.

The parity equations are obtained using a least-squares approach described in ref. 10. A different $V$ matrix is required for each configuration of sensors which remains after a failure is detected and isolated. The parity equations for the full set of sensors is

$$\begin{aligned} p_{a_1} &= 0.7906\delta s_{a_{x1}} - 0.1581\delta s_{a_{x2}} - 0.4320\delta s_{a_{y2}} + 0.1581\delta s_{a_{x3}} - 0.3162\delta s_{a_{y3}} - 0.1581\delta s_{a_{x4}} + 0.1158\delta s_{a_{y4}} \\ p_{a_2} &= 0.7906\delta s_{a_{y1}} + 0.1158\delta s_{a_{x2}} - 0.1581\delta s_{a_{y2}} - 0.3162\delta s_{a_{x3}} - 0.1581\delta s_{a_{y3}} - 0.4320\delta s_{a_{x4}} + 0.1581\delta s_{a_{y4}} \\ p_{a_3} &= 0.7659\delta s_{a_{x2}} - 0.0653\delta s_{a_{y2}} - 0.0828\delta s_{a_{x3}} - 0.5351\delta s_{a_{y3}} + 0.1959\delta s_{a_{x4}} - 0.2786\delta s_{a_{y4}} \\ p_{a_4} &= 0.6396\delta s_{a_{y2}} + 0.1632\delta s_{a_{x3}} - 0.4245\delta s_{a_{y3}} - 0.5844\delta s_{a_{x4}} + 0.2061\delta s_{a_{y4}} \\ p_{a_5} &= 0.6830\delta s_{a_{x3}} + 0.1830\delta s_{a_{y3}} - 0.1830\delta s_{a_{x4}} - 0.6830\delta s_{a_{y4}} \end{aligned} \tag{14}$$

A GLT formulation of the detection and isolation problems has been developed. Assuming single-axis failures initially, the GLT decision functions for detection and isolation are

$$\begin{aligned} DF_D &= p^T p \\ DF_{I_j} &= \frac{\left(p^T v_j\right)^2}{v_j^T v_j} \qquad j = 1, 2, \ldots, n \end{aligned} \tag{15}$$

These decision functions are strictly functions of the parity equation residuals. The detection decision is made by comparing $DF_D$, which is the sum of the squares of the parity residuals, to a detection threshold. A sensor failure results in a change in the mean value of a sensor output, the parity equation residuals, and the failure detection function. The isolation decision is then made by determining $\max_j(DF_{I_j})$. The value of $j$ that maximizes $DF_{I_j}$ identifies the sensor most likely to have failed.

Since TDOF sensors are used, the isolation of a failed sensor rather than a failed axis requires testing only $n/2$ hypotheses. The decision function for isolation is then simplified to

$$DF_{I_j} = p^T v_j \left(v_j^T v_j\right)^{-1} v_j^T p \qquad j = 1, 2, \ldots, n/2 \tag{16}$$

where $v_j = (v_{2j-1}, v_{2j})$ and $v_{2j-1}, v_{2j}$, are the two columns of the $V$ matrix associated with TDOF sensor $j$. The sensor most likely to have failed is determined by the $j$ which maximizes the isolation function.

The thresholds used with the GLT algorithm are generated similar to the EVT algorithm. That is, to detect flight control level failures, the parity residuals are compared to a constant threshold, and to detect low level sensor failures, the FDI thresholds must be able to accommodate the effects of maneuvering flight. An estimate for maximum sensor error is generated from an analytic expression for the upper bound of the sensor errors where only first order effects including bias, misalignment, and scale factor errors are considered. An upper bound for the GLT accelerometer parity residuals arising from analytical determination of sensor error is,

$$p_{i_m} = \sum_j |v_{ij}| \delta s_{a_m} \tag{17}$$

The analytical part of the threshold is obtained by summing the squares of the upper bound for each parity equation. The resulting expression is

$$T_{a_a} = \sum_{i=1}^{n-3} \left(p_{i_m}\right)^2 \tag{18}$$

The third part of the accelerometer threshold is formed from the sum of the squares of the washout filtered parity residuals. The expression for the total accelerometer threshold is

$$T_a = T_{a_q} + T_{a_a} + T_{a_w} \tag{19}$$

When a failure is detected this expression is modified to account for the reduced number of sensors.

## 6. FLIGHT TEST EQUIPMENT AND PARAMETERS

Figure 6 shows the aircraft installation of the equipment rack built to flight test the FDI algorithms. Total weight for the rack and all equipment is 775 pounds. The RSDIMU consists of two full ATR boxes containing instrument control and compensation electronics and a full ATR box for input power conditioning. The sensor cluster is mounted in the middle row of the pallet.

With the exception of the power distribution panel, the remaining equipment makes up the flight computer system and consists of two 16-bit general purpose mini-computers, two computer control panels, four computer interface chasses, a 1/2 inch cartridge tape recorder, a serial line printer, a hand held terminal, and an RSDIMU Control/Display Panel.

The parameters used for the EVT and GLT threshold tests are given in Table 2. Because of computer timing considerations, a first-order filter was implemented for the soft-level channel. The first-order filter time constants were chosen to reduce the effective noise levels to an acceptable level to enhance failure detection sensitivity. The washout filter time constants were obtained from laboratory tests to minimize false alarms and failure detection times. The constant threshold levels were obtained from sensor bias considerations and to ensure no occurrences of false alarms.

Table 2. EVT and GLT threshold values and time constants.

| | |
|---|---|
| **EVT** | |
| Gyro Flight Control Constant | 0.5 deg/sec |
| Gyro Soft-Level Constant | 0.95 deg/hr |
| Gyro Soft Dynamic Error | 2828 ppm |
| Gyro First-Order Filter Time Constant | 60 sec |
| Gyro Washout Filter Time Constant | 180 sec |
| Accelerometer Flight Control Constant | 0.1 g |
| Accelerometer Soft-Level Constant | 0.9 milli-g |
| Accelerometer Soft Dynamic Error | 1414 ppm |
| Accelerometer First Order Filter Time Constant | 10 sec |
| Accelerometer Washout Filter Time Constant | 120 sec |
| **GLT** | |
| Gyro Flight Control Constant | $1800^2$ $(\text{deg/hr})^2$ |
| Gyro Soft-Level Constant | $0.95^2$ $(\text{deg/hr})^2$ |
| Gyro Soft Dynamic Error | $1.98 \times 10^{-5}$ ppm |
| Gyro First-Order Filter Time Constant | 60 sec |
| Gyro Washout Filter Time Constant | 180 sec |
| Accelerometer Flight Control Constant | $0.1^2$ $g^2$ |
| Accelerometer Soft-Level Constant | $0.9^2(\text{milli-g})^2$ |
| Accelerometer Soft Dynamic Error | $5 \times 10^{-6}$ ppm |
| Accelerometer First Order Filter Time Constant | 10 sec |
| Accelerometer Washout Filter Time Constant | 120 sec |

Four flights were used to test the FDI capabilities of the EVT and GLT algorithms (refs. 11, 12). A Lockheed Electra, shown in figure 7, was used for flight 1 for which the pallet was located approximately 20 ft forward of the cg. For the remaining flights a P-3 Orion was used. For this aircraft the pallet was located approximately at the cg. All flights originated at NASA Wallops Flight Facility. Typically the flights were constant altitude, constant speed, box patterns of sufficient duration to exercise the FDI algorithms. Some periods of moderate turbulence were experienced during the four flights but the conditions mainly consisted of light turbulence. Table 3 gives a brief description of the significant events for flight 3.

Table 3. Time history of events for flight 3.

| Time, sec | Event | Comments |
|---|---|---|
| 627 | Begin takeoff roll Runway 22 | Gyros 3 and 4 failed simultaneously 15 seconds after roll starts |
| 653 | Liftoff-Turn to 210 degree heading | |
| 986 | Right turn to 50 degree heading | Cape Charles VOR |
| 1216 | Left turn to 350 degee heading | Constant altitude 6000 ft. 200 knots airspeed |
| 1300 | | Accels. 3 and 4 failed simultaneously |
| 1372 | Landing gear down | |
| 1400 | Left turn to 210 degree heading | |
| 1491 | Flaps down | |
| 1540 | Touchdown Runway 22 | |

## 7. FLIGHT TEST RESULTS

### 7.1 Edge Vector Test Method

The flight test results for the EVT algorithm from flight 3 are given in figures 8-14. The plots begin 2 seconds before takeoff roll and are terminated prior to touchdown at 1525 seconds. Failures were added to gyros 3 and 4 during the takeoff roll. Failures were added to accelerometers 3 and 4 at 1300 seconds.

The unfiltered gyro EVT parity residuals for gyros 1, 2, and 3 ($p_{g_{12_h}}$, $p_{g_{13_h}}$, and $p_{g_{23_h}}$), are given in figure 8. Since a hard failure (−0.7 deg/sec) was applied to both axes of gyro 4 during the takeoff roll, the residuals for the gyro 4 parity equations are not shown. The effects of the low-level failure applied to gyro 3 are not evident in the unfiltered residuals. The $p_{g_{12_h}}$ residual is not affected by either of these failures. The peak-to-peak variation in the $p_{g_{12_h}}$ residual is normal for these sensors.

The filtered EVT gyro parity residuals are given in figure 9. Only those parity equations unaffected by the hard failure are shown. The effects of the low-level failure added to gyro 3 at 640 seconds is evident in residuals $p_{g_{13_s}}$ and $p_{g_{23_s}}$. The advantage of filtering the gyro residuals is apparent in $p_{g_{12_s}}$, the only parity residual unaffected by the failures. This residual is well behaved and it illustrates the sensitivity with which low failure levels could be detected. The effects of the turns beginning at about 1000 and 1400 seconds are evident in all the filtered gyro parity residuals.

Figure 10 shows the washout compensation added to the thresholds for $p_{g_{12_s}}$, $p_{g_{13_s}}$, and $p_{g_{23_s}}$. It can be seen that the washout filter tends to follow the high frequency excursions of the parity residuals, thus providing false-alarm protection for the FDI process.

The gyro threshold tests for the filtered gyro parity residuals are shown in figure 11. The results of these tests indicates positive results for $(|p_{g_{13_s}}| - T_{g_{13}})$ and $(|p_{g_{23_s}}| - T_{g_{23}})$; this indicates failure detection for those parity equations. The large excursions in the threshold test results indicate that the values used in Table 2 for the threshold terms, particularly misalignment and scale factor error estimates, were too large.

The unfiltered accelerometer parity residuals for flight 3 are given in figure 12. The effects of the dual axis failure on accelerometer 3 (6 milli-g $x$-axis and -1.5 milli-g $y$-axis) at 1300 seconds is evident in $p_{a_{13_h}}$. Analysis of the EVT parity equations shows that the applied failure level will not be detected in either $p_{a_{23_h}}$ or $p_{a_{23_s}}$ (ref. 5).

Figure 13 shows the filtered residuals for accelerometer parity equations $p_{a_{12_s}}$, $p_{a_{13_s}}$, and $p_{a_{23_s}}$. The effects of takeoff and turn dynamics is particularly evident in the $p_{a_{12_s}}$ residual since it contains no failure level. This residual is well behaved and it illustrates the sensitivity with which low-level failures could be detected during straight and level flight.

Figure 14 shows the EVT accelerometer threshold tests for the filtered residuals (the EVT accelerometer washout filtered residuals were not recorded during the flight tests). Since $(|p_{a_{12_s}}| - T_a < 0)$, no failure was detected for this parity equation. The test for $p_{a_{13_s}}$ indicates failure detection with its positive result. As stated above there was no failure detection for $p_{a_{23_s}}$.

7.2 Generalized Likelihood Test Method

The unfiltered and filtered gyro parity residuals for the GLT algorithm are shown in figures 15 and 16. Only three parity residuals are shown for the hard- and soft-level residuals, since $p_{g_4}$ and $p_{g_5}$, the remaining two residuals, were affected by hard failures on the takeoff roll. The unfiltered gyro parity residuals, $p_{g_{1h}}$, $p_{g_{2h}}$, and $p_{g_{3h}}$ show the effects of takeoff dynamics and the GLT algorithm reconfiguration after failure detection and isolation. That is, when the failure in gyro 4 is detected and isolated, the GLT parity Eq.(19) and isolation functions Eqs.(20, 21) are reconfigured to reflect the three remaining instruments. When the second failure is isolated, the system again reconfigures itself to reflect the two remaining instruments (one parity equation). The soft-level parity residuals in figure 16 show the effects of this reconfiguration after the failure detection and isolation. The soft-level parity residual, $p_{g_{1s}}$, remaining after the second isolation shows the effects of turn dynamics.

Figure 17 shows the GLT gyro decision function, $DF_{D_g}$, the washout threshold compensation term $T_{gw}$, and the threshold test, $DF_{D_g} - T_g$. The washout filtered threshold term illustrates the tendency toward false-alarm free operation of the algorithm since high frequency terms appearing in the parity residuals are compensated.

The GLT unfiltered and filtered accelerometer parity residuals are given in figures 18 and 19. These residuals show the effects of takeoff dynamics and reconfiguration after failure detection and isolation. For this flight the GLT algorithm provided a false isolation for the low-level failure applied to accelerometer 3 at 1300 seconds.

7.3 Failure Detection and Isolation Performance

Table 4 lists the performance results for the FDI algorithms in flight test. Since the wind conditions were calm for flight 1, there were two identical redundancy management experiments performed. In both experiments for flight 1, five failure levels, including three gyro and two accelerometer, were added to the sensor outputs. At 5040 seconds during flight 1a, both axes of gyro 4 were opened. Since this failure occurred just prior to the start of a maneuver, the failure was detected and properly isolated as soft level failures in two seconds for both the EVT and the GLT algorithms. At 5160 seconds the x-axis of accelerometer 4 a bias level of 10 milli-g was added to the accelerometer output. This failure was detected in 69 seconds and 15 seconds for the EVT and GLT algorithms, respectively. Similarly the next two failures show that the GLT algorithm detects and isolates soft-level failures faster than the EVT algorithm. The flight control level failure added to the y-axis of gyro 2 was detected on cycle for both algorithms.

The same set of failures was repeated in flight 1b. The open failure took longer to detect for this occurrence than flight 1a because the flight was straight and level. The accelerometer failures were detected in times similar to flight 1a. The increase in time to detect the gyro 3 failure is caused by the residual level at the time of application. A flight control level failure was applied to gyro 2 eighteen seconds before touchdown.

For flight 2, both axes of gyro 4 were given flight control level failures during the takeoff roll. Both the GLT and EVT algorithms detected and properly isolated this failure on cycle. Ten seconds later, while still on the takeoff roll, gyro 3 was given a low-level bias failure. The aircraft maneuvers after takeoff caused an increase in the gyro threshold and, hence, a long detection and isolation time for the gyro 3 failure. The FDI for the single-axis failures in accelerometers 4 and 3 are similar to the results of flight 1. Accelerometer 2 was opened during the landing approach; this failure was detected and properly isolated on-cycle.

On the takeoff roll for flight 3, all sensitive axes of gyros 3 and 4 were failed simultaneously. The hard level failure on gyro 4 was detected on-cycle by both algorithms. The soft-level failure on gyro 3 was detected in 251 and 111 seconds by the EVT and GLT algorithms, respectively. At 1300 seconds clock time, all axes of accelerometers 3 and 4 were failed. The flight control level failure on accelerometer 4 was detected and isolated on cycle, but the accelerometer 3 failure was not isolated by the EVT algorithm. Analysis of the EVT parity equations, and hence, the parity residuals shows that $p_{a_{13}}$ will detect the failure, but $p_{a_{23}}$ will not. The GLT algorithm easily detects the failure but produces a false isolation.

The final experiment for the series of flight tests included an open failure on gyro 4 during the takeoff roll. This failure was detected in 8 and 4 seconds for the EVT and GLT algorithms, respectively. At liftoff both axes of gyro 3 were failed. This failure too was detected and properly isolated by both algorithms. Both axes of accelerometers 3 and 4 were failed during the flight and were detected and properly isolated. At 2105 clock time the GLT algorithm indicated a false alarm; this failed IMUB. This failure manifested itself as 1) transient loss of temperature regulation; 2) errors in data transfer between IMUA and IMUB; and, 3) loss of navigation data on IMUB. Post flight analysis indicated flight computer B malfunctioned. Therefore, this was a genuine system failure which was detected. IMUA continued to operate properly for the remainder of the flight.

Table 4. Flight Demonstration of EVT and GLT algorithms.

| Flight | Sensor Failed | Failure Level | Failure Injection Time, sec | FDI Time, sec EVT | FDI Time, sec GLT |
|---|---|---|---|---|---|
| 1a | $s_{g_{x4}}$, $s_{g_{y4}}$ | open | 5040 | 2 | 2 |
| | $s_{a_{z4}}$ | 10 milli-g | 5160 | 69 | 15 |
| | $s_{g_{y3}}$ | 10 deg/hr | 5360 | 70 | 12 |
| | $s_{a_{y3}}$ | 10 milli-g | 5450 | 63 | 15 |
| | $s_{g_{y2}}$ | 2 deg/sec | 5600 | 0 | 0 |
| 1b | $s_{g_{x4}}$, $s_{g_{y4}}$ | open | 770 | 142 | 12 |
| | $s_{a_{z4}}$ | 10 milli-g | 1000 | 71 | 17 |
| | $s_{g_{y3}}$ | 10 deg/hr | 1500 | 156 | 69 |
| | $s_{a_{y3}}$ | 10 milli-g | 1750 | 7 | 14 |
| | $s_{g_{y2}}$ | 2 deg/sec | 3260 | 0 | 0 |
| 2 | $s_{g_{x4}}$, $s_{g_{y4}}$ | 1 deg/sec | 540 | 0 | 0 |
| | $s_{g_{y3}}$ | 2.6 deg/hr | 550 | 700 | 533 |
| | $s_{a_{x4}}$ | 7 milli-g | 1500 | 93 | 23 |
| | $s_{a_{y3}}$ | 7 milli-g | 3500 | 90 | 19 |
| | $s_{a_{x2}}$, $s_{a_{y2}}$ | open | 4650 | 0 | 0 |
| 3 | $s_{g_{x4}}$, $s_{g_{y4}}$ | −0.7 deg/sec | 640 | 0 | 0 |
| | $s_{g_{x3}}$, $s_{g_{y3}}$ | −4 deg/hr, 1 deg/hr | 640 | 251 | 111 |
| | $s_{a_{x4}}$, $s_{a_{y4}}$ | −0.4 g | 1300 | 0 | 0 |
| | $s_{a_{x3}}$, $s_{a_{y3}}$ | 6 milli-g, −1.5 milli-g | 1300 | n.d[1] | 67[2] |
| | | [1]not detected [2]false isolation | | | |
| 4 | $s_{g_{y4}}$ | open | 460 | 8 | 4 |
| | $s_{g_{x3}}$, $s_{g_{y3}}$ | −4 deg/hr, 1 deg/hr | 510 | 132 | 68 |
| | $s_{a_{x4}}$, $s_{a_{y4}}$ | 7 milli-g, −2 milli-g | 700 | 66 | 55 |
| | $s_{a_{x3}}$, $s_{a_{y3}}$ | 2 milli-g, −9 milli-g | 1300 | 55 | 14 |
| | IMUB Computer malfunction resulted in GLT Shutdown | | | | |

## 8. CONCLUDING REMARKS

Two algorithms for failure detection and isolation of a skewed array of collocated TDOF inertial sensors have been flight tested and compared. The algorithms both provide timely detection and isolation of a variety of sensor failures which affect flight control and navigation accuracies. Dual fail-operational capability for the RSDIMU was demonstrated by virtue of third failure detectability. The GLT algorithm demonstrated faster response to sensor failures than the EVT algorithm. The GLT algorithm, however, suffered a false isolation.

## 9. REFERENCES


1. Anon: Preliminary Design of a Redundant Strapdown Inertial Unit Using Two Degree-of-Freedom Tuned Gimbal Gyroscopes, NASA CR-145035, October 1976.

2. Morrell, F. R.; and Russell, J.: Design of a Developmental Dual Fail-Operational Redundant Strapdown Inertial Measurement Unit, Proceedings of NAECON'80, Dayton, Ohio, May 1980.

3. Motyka, P.; Landey, M.; and McKern, R.: Failure Detection and Isolation Analysis of a Redundant Strapdown Inertial Measurement Unit, NASA CR-165658, February 1981.

4. Gai, E.; Harrison, J.; and Daly, K.: Generalized Likelihood Test for FDI in Redundant Sensor Configurations, AIAA Journal of Guidance and Control, Vol. 2, No.1, January-February 1979.

5. Morrell, F. R.; and Bailey, M. L.: A Vector-Based Failure Detection and Isolation Algorithm for a Dual Fail-Operational Strapdown Inertial Measurement Unit, NASA TM-100493, September 1987.

6. Baum, R.; Morrison, E.; and Peters, R.: A Redundant Inertial Strapdown System for IUS, AGARD Conference Proceedings No. 272, August 1979.

7. Bryant, W. H.; Morrell, F. R.; and Bailey, M. L.: Flight Test Configuration for Verifying Inertial Sensor Redundancy Management Techniques, AIAA Paper No. 84-2496, November 1984.

8. Motyka, P.: Reliability Analysis and Fault Tolerant System Development for a Redundant Strapdown Inertial Measurement Unit, NASA CR-166050, March 1983.

9. Craig, R. J.; and Russell, J.: Failure Modes and Redundancy Analysis for the Multifunction Inertial Reference Assembly (MIRA), AFFDL TR-78-25, March 1978.

10. Potter, J.; and Suman, M.: "Thresholdless Redundancy Management with Arrays of Inertial Instruments," Integrity in Electronic Flight Control Systems, Agardograph-224, 1977.

11. Morrell, F. R.; Bailey, M. L.; and Motyka, P. R.: Flight Test Results of a Vector-Based Failure Detection and Isolation Algorithm for a Redundant Strapdown Inertial Measurement Unit, AIAA 4th Flight Test Conference, AIAA-88-2172, May 1988.

12. Morrell, F. R.; Bailey, M. L.; and Motyka, P. R.: Flight Demonstration of Redundancy Management Algorithms for a Skewed Array of Inertial Sensors, AIAA/AHS/ASEE Aircraft Design and Operations Meeting, AIAA-88-4434, September 1988.

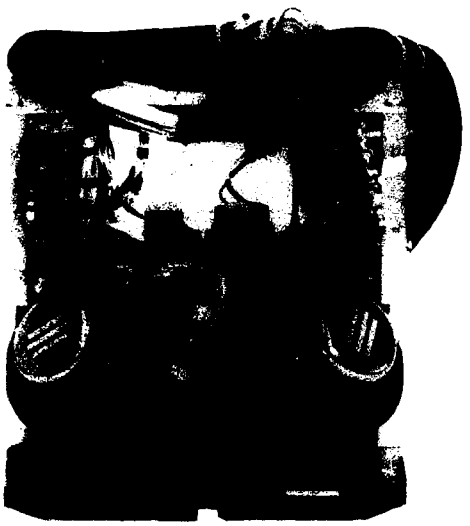


Figure 1.-Skewed inertial sensor cluster.

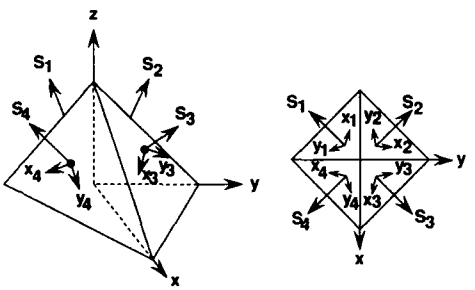



Figure 2.-Skewed inertial sensor geometry.

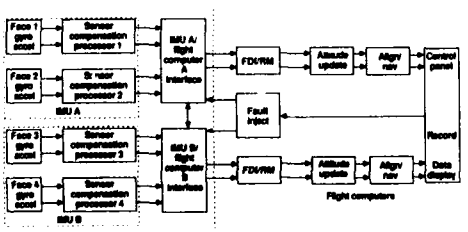


Figure 3.-RSDIMU functional block diagram.

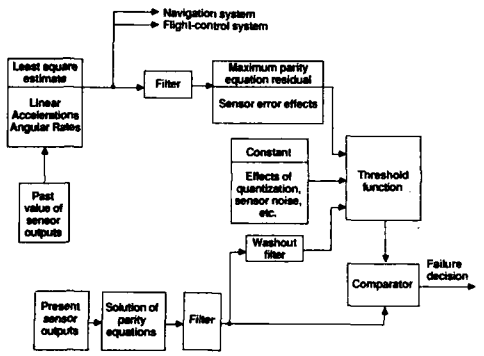



Figure 4.-Block diagram of failure detection process.

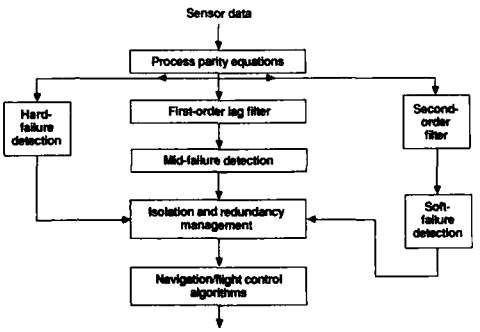



Figure 5.-FDI algorithm multi-level structure.

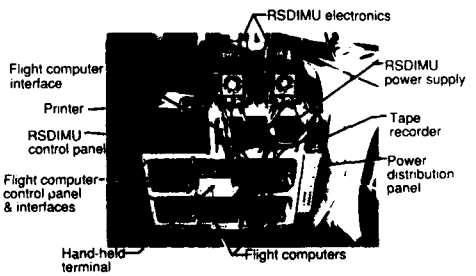



Figure 6.-RSDIMU flight test equipment rack.

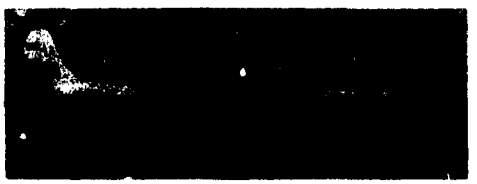


Figure 7.-Lockheed electra flight test aircraft.

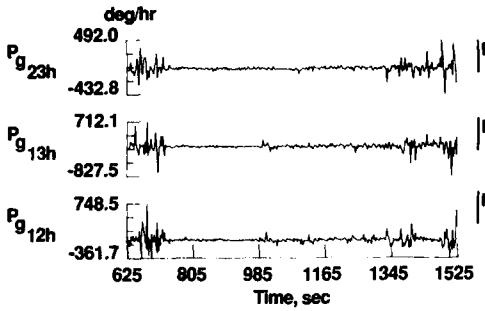



Figure 8.-Flight 3 unfiltered gyro EVT parity residuals.

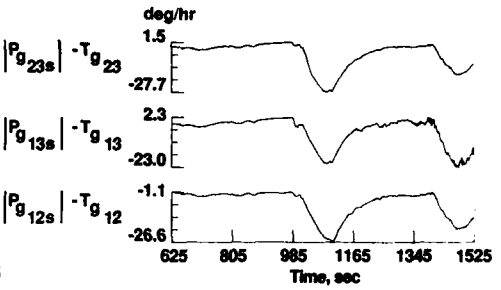



Figure 11.-Threshold tests for filtered gyro parity residuals.

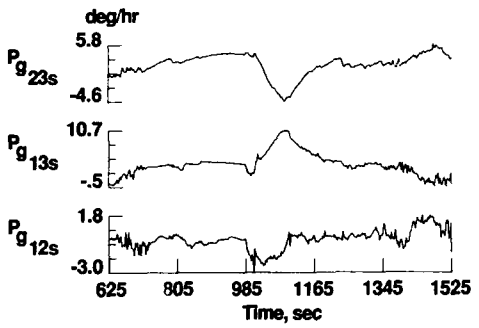



Figure 9.-Flight 3 filtered gyro EVT parity residuals.

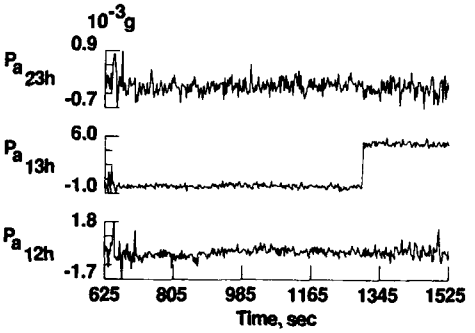



Figure 12.-Flight 3 unfiltered accelerometer EVT residuals.

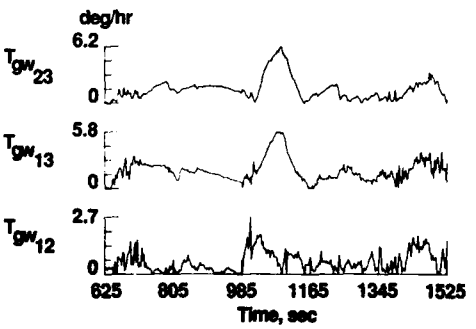



Figure 10.-High frequency gyro threshold compensation.

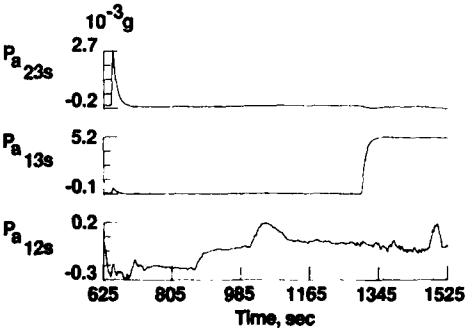



Figure 13.-Flight 3 filtered accelerometer EVT residuals.

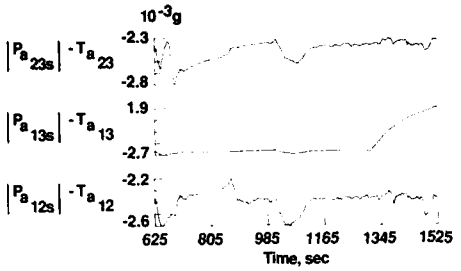



Figure 14.-Threshold tests for filtered accelerometer residuals.

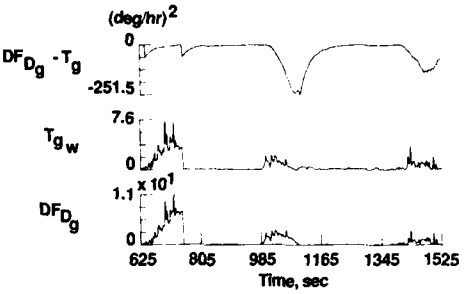



Figure 17.- GLT gyro decision function, high frequency threshold compensation, and threshold test results.

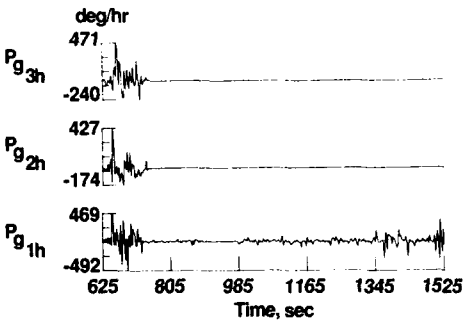



Figure 15.- Flight 3 unfiltered gyro GLT parity residuals.

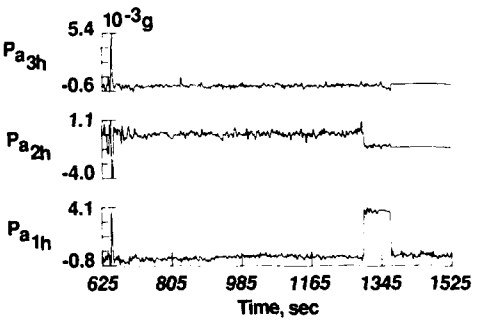



Figure 18.-Flight 3 unfiltered accelerometer GLT parity residuals.

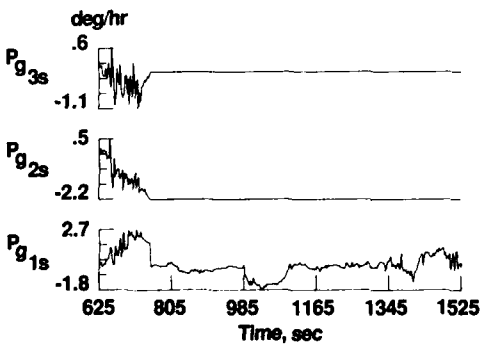



Figure 16.-Flight 3 filtered gyro GLT parity residuals.

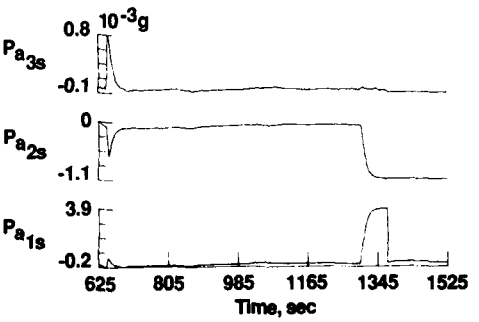



Figure 19.-Flight 3 filtered accelerometer GLT parity residuals.

# FLIGHT DEMONSTRATION OF A SELF REPAIRING FLIGHT CONTROL SYSTEM IN A NASA F-15 FIGHTER AIRCRAFT

by

**James M.Urnes**
McDonnell Aircraft Company
McDonnell Douglas Corporation
St. Louis, Missouri
United States

**James Stewart**
NASA Ames Research Center
Dryden Flight Research Facility
Edwards, California
United States

**Captain Robert Eslinger**
Wright Research and Development Center
Wright-Patterson AFB, Ohio 45433-6553
United States

## 1.0 INTRODUCTION

Battle damage causing loss of control capability can compromise mission objectives and even result in loss of the aircraft. The Self Repairing Flight Control System (SRFCS) flight development program directly addresses this issue with a flight control system design that can measure the damage and immediately refine the control system commands to preserve mission potential. Furthermore, the system diagnostics process can detect in flight the type of faults that are difficult to isolate post flight, and thus cause excessive ground maintenance time and cost. The SRFCS diagnostics feature enhances aircraft alert readiness by reducing ground servicing time.

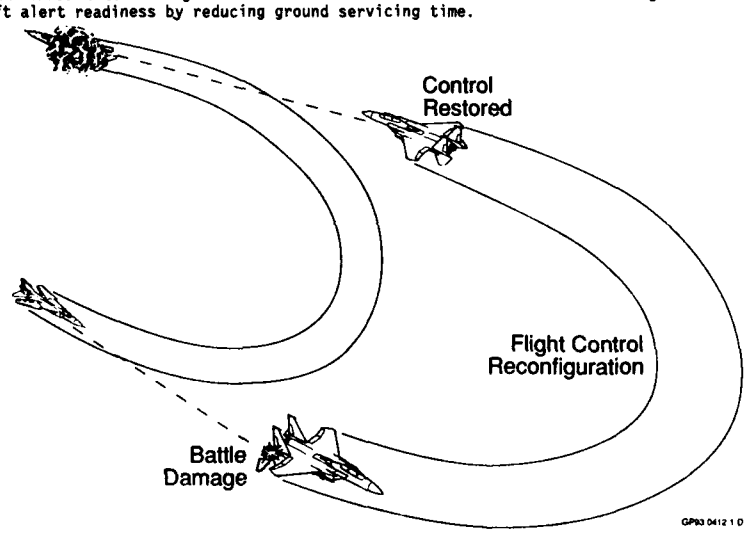



**Figure 1. SRFCS Preserves the Combat Capability of the Damaged Fighter Aircraft**

The control systems of today's fighter aircraft have the control power and surface displacement to maneuver the aircraft in a very large flight envelope with a wide variation in airspeed and g maneuvering conditions, with surplus force capacity available from each control surface. Digital flight control processors are designed to include built-in status of the control system components, as well as sensor information on aircraft control maneuver commands and response. In the event of failure or loss of a control surface, the SRFCS utilizes this capability to reconfigure control commands to the remaining control surfaces, thus preserving maneuvering response. This reconfiguration system permits the damaged aircraft to continue the mission (Figure 1) and return to base. Damage detection must be fast and accurate, and reconfiguration must restore sufficient flight response.

Correct post-flight repair is the key to low maintainability support costs and high aircraft mission readiness. The SRFCS utilizes the large data base available with digital flight control systems to diagnose faults. Built-in-test data and sensor data are used as inputs to an Onboard Expert System process to accurately identify failed components for post-flight maintenance action. This diagnostic technique has the advantage of functioning during flight, and so is especially useful in identifying intermittent faults that are present only during maneuver g loads or high hydraulic flow requirements. These faults are very difficult to isolate in current post-flight maintenance, resulting in Can Not Duplicate (CND) inconclusive ground checks and excessive support man-hours.

McDonnell Aircraft Company working with subcontractor General Electric Controls Division has developed a flight system to test the reconfiguration and onboard maintenance diagnostics concepts on a NASA F-15 fighter aircraft. Key objectives are:

- Flight evaluate a control reconfiguration strategy with three types of control surface failure
- Evaluate a cockpit display that will inform the pilot of the maneuvering capacity of the damaged aircraft
- Flight evaluate the Onboard Expert System maintenance diagnostics process using representative faults set to occur only under maneuvering conditions
- Determine software requirements to install the system in a digital flight control system

The flight development program is sponsored by the USAF Wright Research and Development Center, with flight testing by the NASA Dryden Flight Research Facility.

## 2.0 SRFCS TECHNOLOGIES AND BENEFITS

In realtime reconfiguration, the flight control system automatically redistributes control power among remaining force and moment effectors following battle damage or component failures. Reconfiguration dampens impairment transients, retains stability, reduces pilot work load, and maximizes remaining control performance. Reconfiguration improves survivability and can also be used to improve flight control system reliability. With reconfiguration, the control surfaces provide a new level of aerodynamic redundancy which allows designers to reduce component complexity and levels of hardware redundancy of the control system components. For example, designers can simplify the actuators and hydraulic system by placing single channel simplex actuators on certain control surfaces. Compensation of simplex actuator failure modes then depends on the reconfiguration of the remaining control surfaces instead of hardware redundancy within the actuator. Simplification of flight control system hardware through use of reconfiguration improves the reliability and maintainability of the flight control system while reducing weight and life cycle costs.

The heart of reconfiguration is a Failure Detection, Isolation, and Estimation (FDIE) algorithm. FDIE is needed to: (1) determine that there is an impairment; (2) where on the aircraft it occurred; and, (3) what type of impairment it is. Local FDIE looks for impairments of specific components of the aircraft. For example the command to an actuator is compared, through a model of the actuator, to the surface response. A miscompare over a certain threshold and time window would lead to the conclusion that the actuator is failed. Global (or aircraft path) FDIE monitors overall aircraft response and is used for battle damage cases, which can include missing control surfaces, fuselage damage and wing damage. Global FDIE uses flight control system sensors to compare measured aircraft response to a modeled response. Failure or damage signatures emerge from comparison residuals that are then used to isolate and estimate the impairment. The F-15 program includes control surface test failures that require both local and global FDIE.

The reconfiguration system uses FDIE information to redistribute control authority among remaining control surfaces. FDIE provides impairment information to a constrained pseudo-inverse, called a pseudo-surface resolver or a control mixer, which reconfigures the control surface gains. The pseudo-inverse technique determines the changes in control surface deflections that are required to reconstruct unimpaired forces and moments. The pseudo-inverse process is constrained to limit commands to less effective surfaces and to handle saturation of actuator limits.

A Positive Pilot Alert (PPA) system also uses FDIE information to tell the pilot what happened and what are the new performance and mission constraints. Immediate performance information is displayed on the HUD, and the pilot can call up additional information on a multipurpose display. PPA also provides emergency procedures and constraints on future mission segments. For example, PPA can help the pilot safely fly a damaged aircraft while up-and-away, but PPA also provides landing information to aid in deciding if diversion to another base is necessary. Reference (1) discusses an example PPA design.

The SRFCS Maintenance Diagnostics system operates during flight and parallels the function of the reconfiguration systems' FDIE. Both technologies detect and isolate flight control system faults. However, the diagnostics system can include a larger number of system components and uses a much larger data base to identify faulty Line Replaceable Units. An expert system technology is utilized, consisting of a knowledge base and an inferencing engine that is programmed in the flight control system digital processors. The knowledge base is organized in rule relationships that will efficiently isolate the failure using the inferencing process. The rules are determined by experienced maintenance technicians using flight control system technical descriptions.

The SRFCS onboard expert system is designed to isolate faults to the LRU level during the return flight to base, with the necessary repair data transmitted after the flight to ground maintenance personnel via a lap-top computer and display unit that the ground crew couples to the aircraft. This computer uses expert system diagnostics to complete the isolation of failures that cannot be totally identified in flight, such as a wiring problem involving many bulkhead connectors. The expert system quickly guides technicians through the most efficient maintenance process to find the fault.

An airborne expert system can virtually eliminate CND maintenance codes, which are faults that appear in the flight environment, but cannot be duplicated on the ground. For example, a pin in a connector can come loose during a high G maneuver, but reconnect during 1 G level flight. This fault is extremely difficult to diagnose on the ground, and leads to unnecessary removals of operational components. RETOKs (retest OKs) occur when the operational components are sent to an intermediate shop or depot, tested, found to be non-faulty and returned to the flight line. CNDs and RETOKs are expensive and contribute to aircraft down time. The airborne diagnostic incorporates maintenance procedure expert system rules together with data from flight control system sensors and bus signals operating over a moving window of time. When faults occur, the system uses this data to immediately begin the inferencing process while the symptoms are present, thus eliminating the majority of CND activity.

The combination of reconfiguration and expert system maintenance diagnostics provides greater survivability, greater reliability and more efficient maintenance. These benefits lead to lower life cycle cost in peace-time with greater aircraft availability in wartime.

## 3.0 SRFCS PROGRAMS RELATING TO THE NASA F-15 FLIGHT TEST

Figure 2 shows the schedule and milestones for the SRFCS program, which include feasibility and pilot simulator airframe studies, designs of reconfiguration and maintenance diagnostics systems which are used in the F-15 SRFCS flight test aircraft.

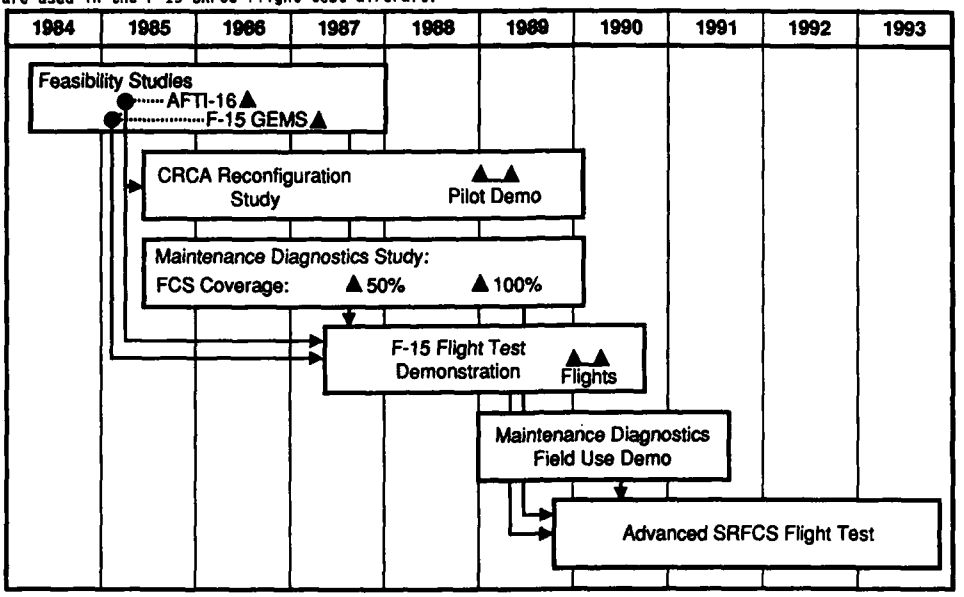



**Figure 2. SRFCS Development Program**

### 3.1 REAL-TIME RECONFIGURATION DEVELOPMENT

A limited flight envelope reconfiguration system was designed for the Flight Dynamics Laboratory's AFTI/F-16 aircraft (Reference 2). This study determined the feasibility of applying reconfiguration algorithms to fighter aircraft. A Sequential Probability Ratio Test (SPRT) was the basis for the FDIE algorithm. This first generation reconfiguration system, called the control mixer, was tested in the USAF Large Amplitude Multi-mode Aerospace Research Simulator (LAMARS). This pilot-in-the-loop, motion-base simulation demonstrated the feasibility of the reconfiguration algorithms. The algorithms were then successfully flown on a Total In-Flight Simulator aircraft (a modified C-131H). These tests proved that algorithms can detect, isolate and estimate aircraft impairments without false alarms, and that reconfiguration restores flight performance to the damaged aircraft. The LAMARS simulation also indicated the need for a PPA system to help pilots maintain safe flight following reconfiguration. These reconfiguration algorithms and display concepts are the baseline for the F-15 flight test demonstration.

Based on the early success of the feasibility study, another study was initiated (References 1, 3, 4 and 5) to develop second generation reconfiguration algorithms for a Control Reconfigurable Combat Aircraft (CRCA), shown in Figure 3. This aircraft was considered a reasonable baseline for future fighter aircraft. The CRCA aerodynamic model is based on extensive wind tunnel testing. The CRCA reconfiguration study expanded the flight and maneuver envelope of the AFTI-16 feasibility study and used m-ary (versus binary) hypothesis testing for FDIE. M-ary hypothesis testing serves the same purpose as the feasibility study FDIE, but it can use a less accurate (smaller) aircraft model because it is more tolerant of model errors. The CRCA study also concentrated on potential impacts reconfiguration can have on aerodynamic design and on flight control system architectures. Examples of architecture impacts include reduced levels of computer and sensor redundancy and the use of simplex versus dual tandem actuators. The reconfiguration and PPA algorithms were flown on a CRCA simulation on LAMARS. The simulation activity examined techniques to eliminate false alarms while minimizing missed detections, incorporation of robust (fault tolerant) control laws with reconfiguration, design of a PPA system, and evaluation of flying qualities following reconfiguration. Reference 6 reviews the SRFCS program.

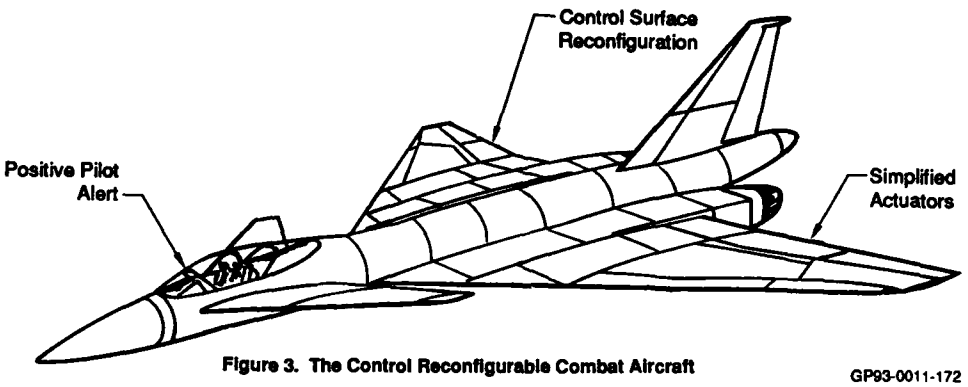



**Figure 3. The Control Reconfigurable Combat Aircraft**

GP93-0011-172

3.2 EXPERT SYSTEM MAINTENANCE DIAGNOSTICS DEVELOPMENT

The initial expert system maintenance diagnostic feasibility study resulted in a personal computer (PC) demonstration system, called the General Electric Maintenance System (GEMS). This was a first look at a rule-based, ground diagnostics system for a limited portion of the F-15 flight control system. Reference 7 discusses GEMS in detail. This rule based expert system concept is updated for airborne application in the F-15 flight test demonstration. Reference 8 further describes the F-15 system.

A second generation Flight Control Maintenance Diagnostics System (FCMDS) was developed for F-16 flight control systems. FCMDS is a frame-based, ground diagnostics system that covers the entire flight control systems, including wiring and connectors. Reference 9 describes the system. It exists on a personal computer and includes user friendly help facilities. FCMDS has been demonstrated at MacDill Air Force Base where engineers updated the system design based on user comments.

4.0 THE OBJECTIVES OF THE F-15 FLIGHT TEST OF THE SRFCS

The major goal of the SRFCS F-15 flight development program is to transition real-time reconfiguration and expert maintenance diagnostics to the operational community. Flight demonstration is an important tool in the technology transition process, and the NASA F-15 flight test is an important, near-term (Fall 1989) transition milestone.

The F-15 SRFCS algorithms were developed during the initial stages of the CRCA study. At that time, the algorithms from the feasibility studies were mature enough to progress to flight test. The F-15 flight test consists of a reconfiguration system, a HUD displayed Positive Pilot Alert system, and an airborne version of expert system diagnostics previously developed for GEMS.

During the F-15 flights the Onboard Expert System (OES) diagnostics will inference on an expanded set of CND scenarios. For the first time, the airborne diagnostic system will be interfaced to a ground (GEMS) system. The OES will inference as far as possible in the air. After the flight, information will be passed from the onboard processor to GEMS to be reported to ground technicians. GEMS then leads the technicians through any remaining troubleshooting.

During the flights, the tie between reconfiguration and maintenance diagnostics will be demonstrated. Both technologies depend on some type of fault detection and isolation. Reconfiguration must occur quickly, while maintenance diagnostics can happen in the background, allowing time for failure isolation. Both systems will operate during a fault scenario that requires real-time reconfiguration to regain performance and airborne diagnostics to explain to the maintenance technicians what happened.

The NASA F-15 flight test is the first demonstration of real-time reconfiguration and diagnostics on a high performance fighter. The flight test will provide answers to important technical questions:

- Will reconfiguration occur fast enough to ensure stability of a fighter aircraft?
- Will the reconfiguration algorithms fit in fighter aircraft flight control processors?
- Can the reconfiguration algorithms perform with existing flight control sensors?
- What is the effect on pilot workload and aircraft target tracking?
- What are the constraints on reconfiguration commands due to aircraft structural properties such as aeroservoelastic interaction?
- How do the fault detection algorithms work in the presence of real sensor noise, air turbulence and modeling error? Can these factors be accommodated to prevent false alarms or inaccurate fault identification?
- Can the types of failures relating to CND or inaccurate maintenance diagnostics be correctly identified with the in-flight expert system approach?

## 5.0 F-15 SRFCS

Figure 4 is a block diagram of the F-15 SRFCS flight control system, which includes the standard mechanical and electronic Control Augmentation System (CAS). The F-15 CAS serves to provide stability augmentation and command response enhancement through control laws implemented in a dual channel digital Electronic Flight Computer. This baseline mode of the system is unchanged until a fault occurs. For the F-15 flight test demonstration, two additional SRFCS commands are added to the CAS servo commands:

- An Impairment Control designed to force the control surface to represent failure conditions. This software module is for flight test only, and not part of a production SRFCS system.

- SRFCS Correction Commands to add the reconfiguration to each control surface servo.

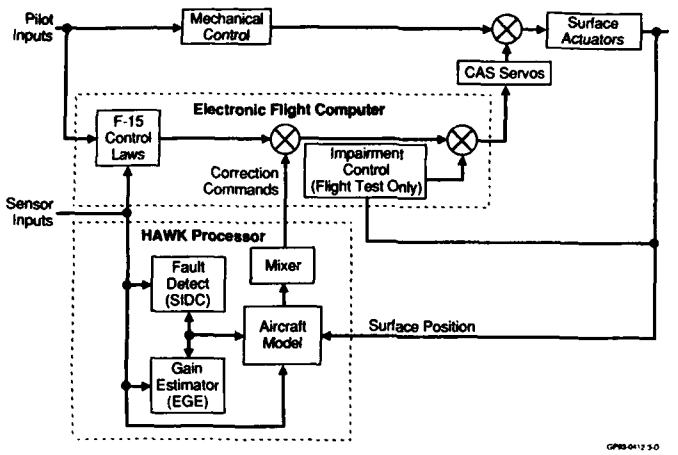



**Figure 4. F-15 SRFCS Reconfiguration**
Flight Test Demonstration

The reconfiguration correction commands are calculated in a high capacity HAWK flight processor, and are derived from four software subroutines:

- Surface Impairment Detection and Classification (SIDC) to detect which surface has failed and the type of impairment.

- Effector Gain Estimator (EGE) to determine how much of the damaged surface remains.

- Aircraft model containing dynamic models of the normal and impaired aircraft.

- Command Mixer that determines the amount of command to transmit to each remaining surface.

The flight test aircraft is configured with three impairments that are selectable from the pilot's station; all three affect the right horizontal stabilator. The impairments are activated with software commands to the stabilator servo actuator to accurately represent the desired failure (Figure 3). The commands negate the mechanical system inputs and pattern the stabilator for the desired impairment. Once the failure type is selected and activated by the pilot, it remains active throughout the fault detection sequence and pilot evaluation of the reconfigured airplane. Both the failure and the correction commands disappear upon pilot deactivation of the reconfiguration test mode through a switch on the control stick.

Three types of stabilator failure are possible with the flight test mechanization shown in Figure 5:

- Fixed Control Surface representing a mechanical jam.

- Trailed Control Surface representing electrical or hydraulic failures. The degree of trail response is controllable through a lag time constant in the software.

- A partially missing surface representing battle damage.

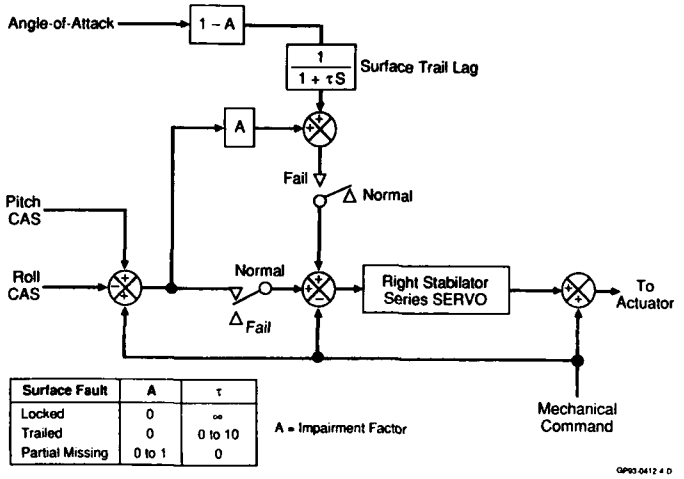



**Figure 5. Control for Partial Surface Loss Model**

The reconfiguration process requires computation of three aircraft acceleration state vectors (Figure 6). The actual aircraft acceleration $X_A$ is continually monitored with a modeled aircraft vector $X_M$ for comparison in the SIDC fault detection. Upon detection of a fault the EGE surface gain estimator functions with the SIDC to provide data to a second aircraft model that acquires the detected impairment. The two models are then used to compute the correction commands which force the modeled impaired aircraft acceleration $X_{IMP}$ to match the unimpaired vector $X_M$. The correction commands also drive the real aircraft, and thus restore the acceleration $X_A$ to be like an undamaged F-15.

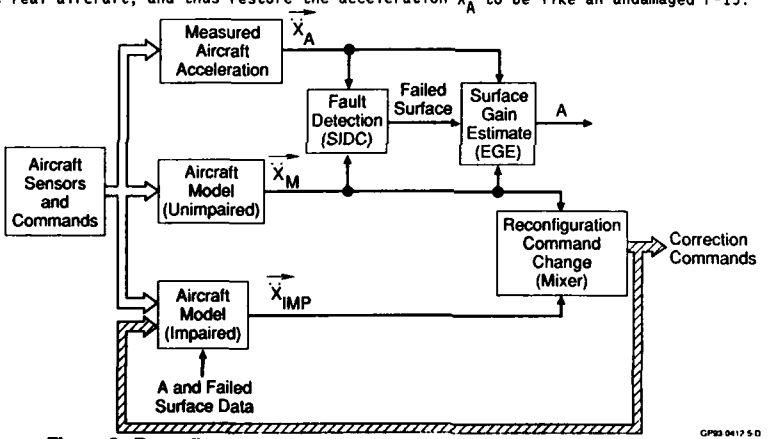



**Figure 6. Reconfiguration Process Incorporates Aircraft Dynamic Models**

The SIDC fault detection scheme shown in Figure 7 continually monitors the flight control system actuator response and accelerations of the aircraft model to determine if errors exist that may be due to failed system components. Figure 4 shows the two paths used in SIDC fault detection. Should actuator commands and outputs not agree, a possible fault will be indicated. Statistical verification with angle of attack information determines if the suspected fault is a locked or trailed actuator. If the actuator response is satisfactory but the aircraft acceleration error persists, then a damaged control surface is declared and a acceleration error vector process is used to identify the specific control surface. This process involves pairwise hypothesis testing performed in Sequential Probability Ratio Tests to identify which aircraft control quadrant is affected and thus identify the failed control surface. Threshold limits are used on the actuator and acceleration error detect computations to prevent false fault activation due to sensor noise or air turbulence.

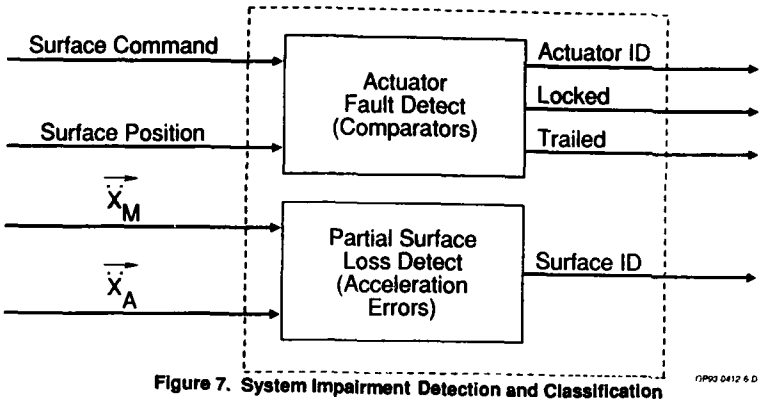



**Figure 7. System Impairment Detection and Classification**

If an actuator fault is declared, the system immediately activates the reconfiguration mixer subroutine, which computes the correction commands to the remaining control actuators to return the aircraft to normal response. This Mixer subroutine uses a pseudo inverse matrix computation as shown in Figure 8. The two aircraft dynamic models contain aerodynamic, mass properties, and control surface representations sufficient to determine the aircraft acceleration vector. Impairments detected by the SIDC are inserted in one model. The other model always remains in the undamaged state. A control derivative matrix is continually calculated from the impaired model (B*) to be used in a pseudo inverse computation of the correction commands. Upon activation of the Mixer, two acceleration vectors are calculated to provide an error vector for the inverse matrix computation. Incremental correction commands are summed and entered in the impaired model, thus driving the acceleration error to zero and restoring the acceleration vector to match the unimpaired acceleration response. The same correction commands are summed with the CAS control surface commands to return the F-15 to normal flight response. A weighting matrix W is used to limit commands to surfaces that are impaired or hardware limited. The correction commands will continually change depending on pilot or feedback sensor commands or change in flight condition airspeed and altitude.

Should the SIDC detect a surface partially missing due to battle damage, a different sequence of reconfiguration occurs in which the EGE estimates the degree of damage and how much control surface span remains available for command. The estimation process incorporates a Kalman filter state estimator and the two aircraft models as shown in Figure 9. When the impaired surface gain has been estimated, the remaining span value (A) is locked in the damaged aircraft model, and the reconfiguration mixer proceeds to calculate the proper correction commands.

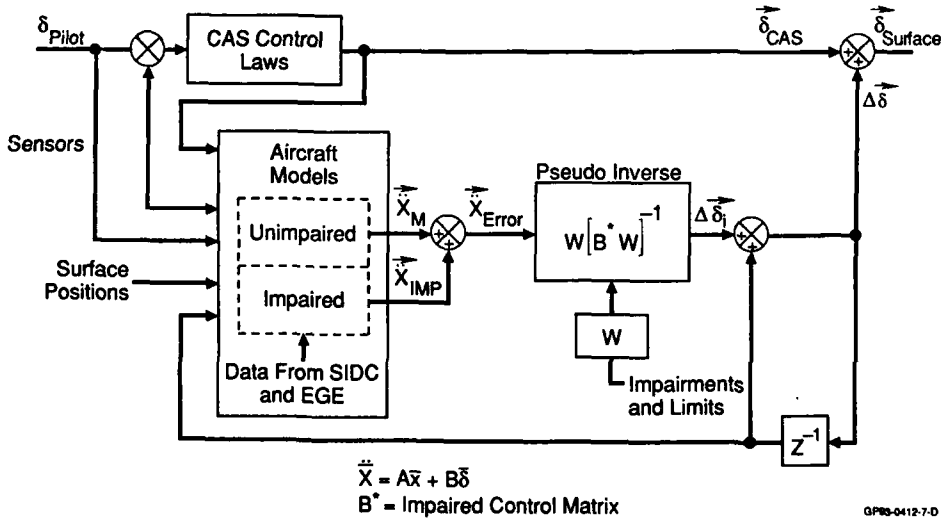



**Figure 8. SRFCS Mixer Restores Control**

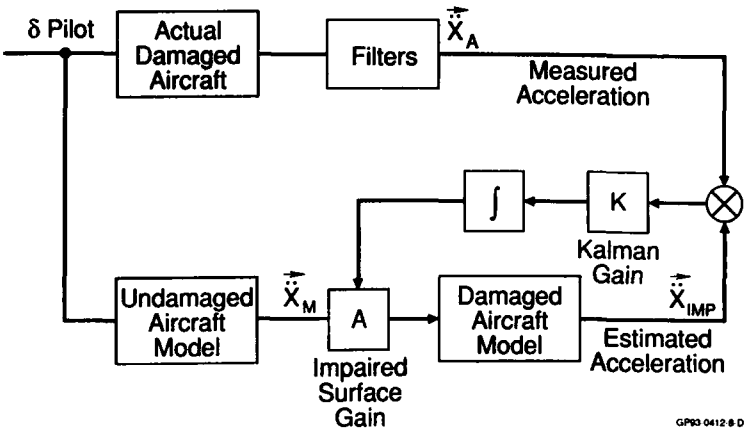



**Figure 9. Estimation Process to Obtain Impaired Surface Gain**

The results of the damage detection process for a right stabilator span that is 80% missing are illustrated in Figure 10. The comparison of the model acceleration and actual aircraft acceleration response is shown in Figure 11 for a maneuver sequence consisting of a 2 g turn followed by the damage to the right stabilator at two seconds. Some modeling error is evident in the yaw acceleration; this is due to differences in yaw aerodynamic drag and induced flow on the vertical stabilizer. The damage impaired model implements actual missing span drag change whereas the aircraft response uses the flight test representation where the horizontal stabilator remains on the aircraft.

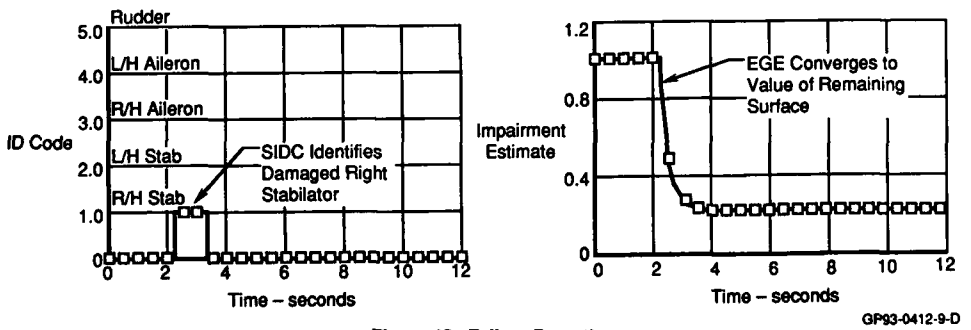



**Figure 10. Failure Detection**
80 Percent Missing Right Stabilator

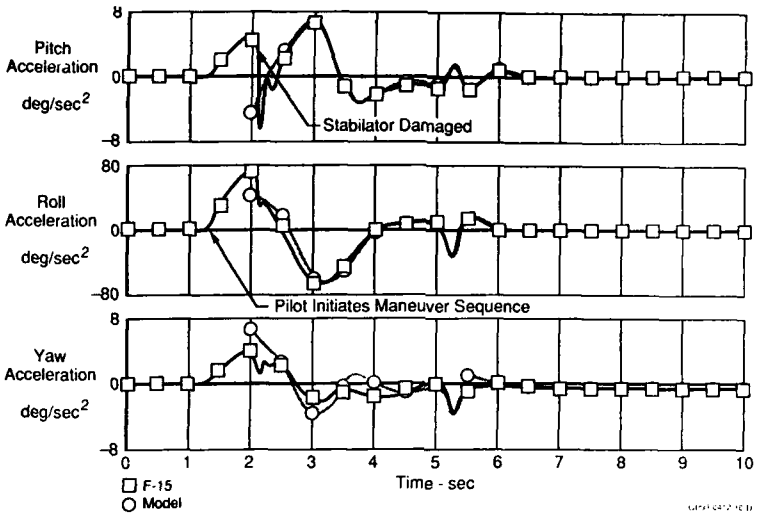



**Figure 11. F-15 and Aircraft Model Response Conpared**
80 Percent Right Stabilator Missing

An example is shown in Figure 12 of the F-15 test aircraft SRFCS software performing the reconfiguration for a battle damaged right stabilator missing 80% of its span. The fault is detected as the pilot initiates a bank maneuver and the reconfiguration engages .35 seconds later. The bank response is maintained very close to the undamaged F-15 response.

The pilot must be made aware of the damage effects on remaining maneuvering capability of the reconfigured fighter. Figure 13 shows the display added to the pilot's Head Up Display (HUD) immediately after the damage detection and reconfiguration occurs. This Positive Pilot Alert warns the pilot of the damage problem and provides positive information on permissible maneuvering capability. The added display is a box containing a maneuver symbol that is positioned by g level in vertical and by roll rate in lateral. The shape of the box is controlled by reconfiguration parameters to maneuver limits available, in this case less capability in left roll due to a failed left stabilator. The pilot maintains the maneuver symbol in the box for safe control of the aircraft.

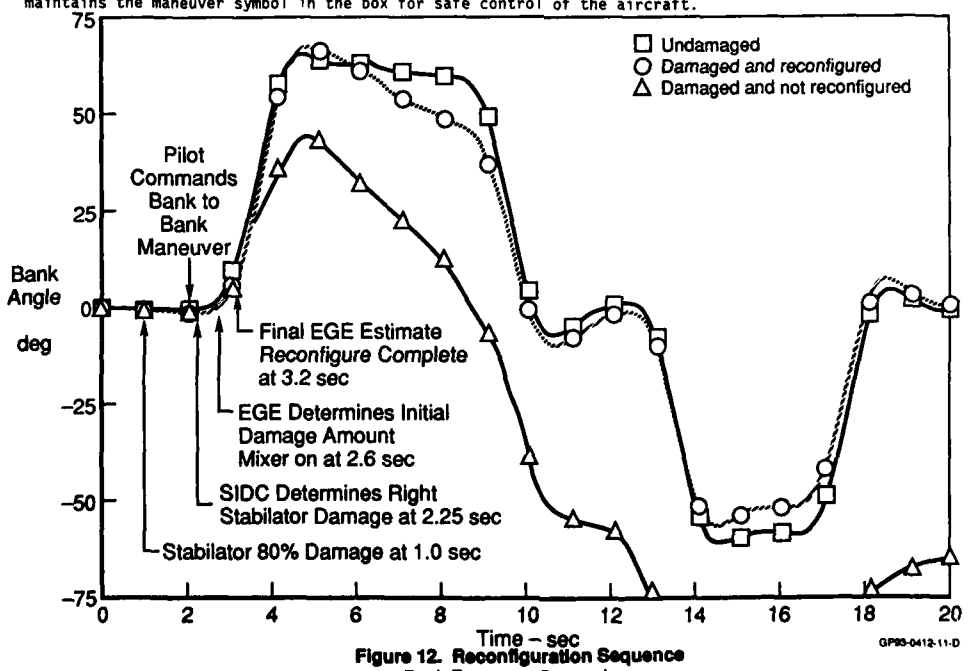



**Figure 12. Reconfiguration Sequence**
Bank Response Comparison

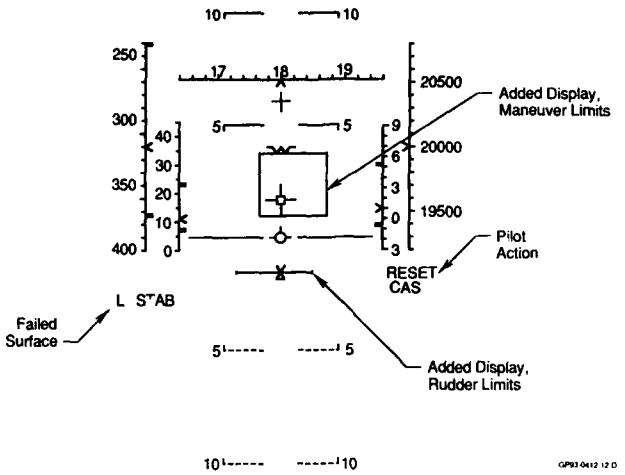



**Figure 13. SRFCS HUD Display With Left Stabilator Problem**

## 6.0 SRFCS ONBOARD EXPERT SYSTEM DIAGNOSTICS DESCRIPTION

The SRFCS Onboard Expert Diagnostics system isolates failures as they happen in flight, using available Built In Test signals and additional sensor information on maneuver conditions and cockpit switch events that are present at the time of the failure. Fact relationships are grouped in rules to find the most likely failure using the expert system forward chaining inferencing process. Figure 14 shows a portion of the rules used in a flight test software mechanization that detected a failure in one of the two angle of attack sensors on the test F-15. A lag or delay was intentionally placed on the right sensor to represent a failure that is difficult to detect post-flight. The fact relationships are connected in rules using logic notation:

Example: Rule 1A

IF Fact 0829 (Pitch CAS Engage) is FALSE
AND Fact 0830 (Roll CAS Engage) is FALSE
AND Fact 0831 (Yaw CAS Engage) is TRUE

THEN FACT 0831 (No Pitch and Roll Disengage) is = FALSE

| Facts | | Rules | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. | Mnemonic | 1A | 2A | 3A | 1B | 2B | 1C | 1D | 1E | 2E | 3E | 4E |
| 0829 | Pitch CAS Engage | F | | | | | | | | | | |
| 0830 | Roll CAS Engage | F | | | | | | | | | | |
| 0831 | Yaw CAS Engage | T | | | | | | | | | | |
| 0813 | No Pitch and Roll Axes Disengage | =F | T | | | | | | | | | |
| 0837 | No Pitch and Yaw Axes Disengage | | T | | F | | | | | | | |
| 0838 | No Roll and Yaw Axes Disengage | | T | | | | | | | | | |
| 0839 | NoPitch+Roll+Yaw Axes Disengage | | T | | | | | | | | | |
| 0802 | No Multiple Axis or Other Faults | | =T | T | | | | | | | | |
| 0835 | No Disengage | | | T | | | | | | | | |
| 0801 | No Single Axis Faults | | | T | | | | | | | | |
| 0800 | CAS Mode System | | | =T | | | | | | | | |
| 0842 | Roll CAS Re-Engage | | | | T | | | | | | | |
| 0841 | Roll Reset Fail After Disengage | | | | =F | F | | | | | | |
| 0550 | Fault Code 2210C1AZ | | | | | =F | =T | | | | | |
| 0570 | ASP Indicator Not Latched | | | | | | T | =T | | | | |
| 0001 | Fault Code 2210A2A1 | | | | | | | T | =T | | | |
| 0041 | AOA Monitor | | | | | | | | T | =T | | |
| 0028 | Right AOA Signal on Bus | | | | | | | | | T | | |
| 0048 | Left AOA Signal on Bus | | | | | | | | | T | | |
| 0027 R | Right AOA Mechanical | | | | | | | | | T | | T |
| 0047 R | Left AOA Mechanical | | | | | | | | | T | T | |
| 0049 | Left AOA Not Lagging | | | | | | | | | | =T | |
| 0029 | Right AOA Not Lagging | | | | | | | | | | | =T |

**Figure 14. Angle-of-Attack Sensor Failure**
**Inferencing Process to Detect a Failed Angle-of-Attack Sensor**

Different inputs than shown to any of Facts 0829, 0830, and 0831 will cause Fact 0831 to be set TRUE. This result can then be used in Rules 2A and 1B for further processing. Facts noted with R are exit or reporting facts. The status of the facts is shown in Figure 14 immediately after the process is completed, with a prime (') notation on the facts triggered due to the fault are shown opposite to the normal, non-failed system. The consequent (=) facts are noted, which chain to the next rule until the correct diagnosis is reached (Fact 0027) that the right angle of attack sensor is failed. The expert system is efficient in that groupings of related events are possible, and logical deductions can be made and causal relationships seen. Additional rules can be easily added without a rework of the main flow of the system.

The F-15 SRFCS flight program evaluates the expert system process using six examples representing CND type faults of mechanical, electrical or hydraulic system. Figure 15 is a table of these test faults. Scenario 2 also makes use of the reconfiguration fault detection (SIDC), applying the stabilator damage detection properties to assist in diagnosing a missing stabilator connecting pin.

The SRFCS software is installed in two processor units on the F-15 test aircraft. Figure 15 illustrates the aircraft flight avionics hardware and the associated support equipment used for system integrated testing. The dual channel Digital Flight Control Computer (DFCC) provides the CAS control sensor and actuator command reconfiguration system interface, together with fault signals used in the expert system diagnostics. The single channel Rolm HAWK is programmed with the expert system inferencing engine, fault logic, reconfiguration SIDC, EGE and Mixer control modules, and the control display information for the pilot. Digital processing incorporates 80 Hz update in the DFCC and 20 Hz Multiplex Bus communication controlled by the Central Computer. Dual channel command limiters are used in the DFCC for all SRFCS Mixer control command paths to protect the SRFCS from large changes in commands due to hardover failures. The software size in the 32 bit Hawk processor is:

Maintenance Diagnostics: 243 K Memory
3.2 ms throughput

Reconfiguration Flight Control: 57 K Memory
20.8 ms throughput

The SRFCS was tested using the flight components integrated in an avionics flight simulator test environment, shown in Figure 16. The integration host SEL computer is programmed with sensor or command test signal sequences and resulting control commands are verified with analysis data. Fault monitoring and isolation is verified using interrupts on power supply or signal interconnect paths. The system is interfaced with a real-time manned flight simulator cockpit for pilot verification of flight response and displays.

| | Maneuver Conditions | Failure Indication Major System | Pilot Action | Subsystem Failed | Cause |
|---|---|---|---|---|---|
| 1 | > 3g | Roll CAS Disengages | Go to 1g and Reset CAS | Dynamic Pressure Sensor | Connector Fails Under g Load |
| 2 | 1g Small Pitch Inputs | None | None | Stab Surface | Actuator Connecting Pin |
| 3 | 1g Small Pitch Inputs | Pitch, Roll CAS Disengage | Reset CAS | Stab Actuator | Hydraulic |
| 4 | 2g Turn | Autopilot Disengage | Go to 1g and Reengage Autopilot | INS | Platform Stabilization Fails Under g Load |
| 5 | 5g Turn | Pitch, Roll CAS Disengage | Go to 1g, and Reset CAS | Pitch Computer | Card A Loose Connection Under g Load |
| 6 | Pull-Up | CAS Disengage | Go to 1g, and Reset CAS | Right AOA Sensor Vane | Excessive Friction in Vane Rotor |

**Figure 15. In-Flight Maintenance Diagnostics Scenarios**

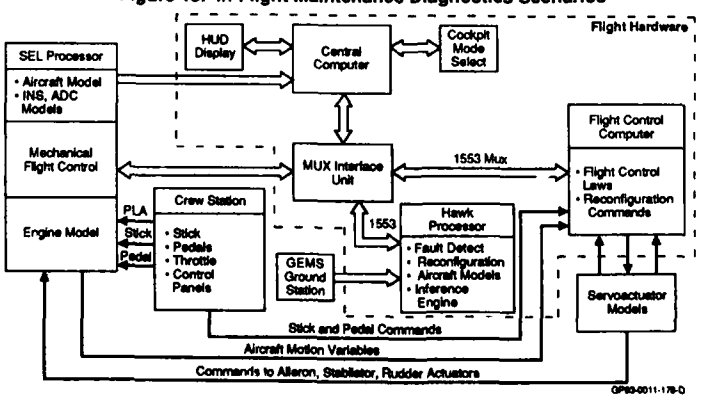



**Figure 16. Flight Control Hardware and Test Facility Simulator**

## 7.0 THE SRFCS FLIGHT TEST PROGRAM

The NASA F-15 airplane to be used for SRFCS flight testing is a national facility for conducting aeronautical flight research. The airplane is highly instrumented and equipped with an integrated digital flight control system.

The aircraft has features that can be used to explore the technologies of digital flight systems, control surface impairments, and controls integration. A large number of unique research experiments have used the on-board computational capability. A summary of results of NASA F-15 flight research programs is listed in Reference 10.

### Specific System Features

Although a number of special features of the NASA F-15 will be used for SRFCS Flight testing, the three most important are the Digital Flight Control Systems, the Avionics System and the Data Acquisition System.

First, the standard F-15 airplane is equipped with a mechanical flight control system that provides control of the ailerons, rudders, and stabilizers. An analog electronics control augmentation system (CAS) operates in all three axis. This standard F-15 arrangement would not be acceptable for SRFCS since it must adapt to changes in the aircraft.

The NASA F-15 airplane has a Digital Electronic Flight Control System (DEFCS) that augments the standard flight control system. The DEFCS replaces the analog CAS. It is a dual-channel, fail-safe system programmed in PASCAL. A military standard 1553 data bus input-output capability with additional capacity in the DEFCS computers will be used for the SRFCS logic. The NASA F-15 is further enhanced for the SRFCS program by the addition of fly-by-wire ailerons.

The F-15 avionics system is shown in Figure 17. Three data buses are used to communicate between the various components. A data bus interface and control unit allows communication between buses.

On the military standard 1553 bus are the Digital Electronic Flight Control System, the NASA data system and a general purpose (Rolm Hawk) digital computer. All of these features are used, allowing a high degree of integration and a sufficient computer and communication capability for implementation of a SRFCS.

The general purpose digital computer uses 32-bit words, has a throughput of approximately 2.5 million instructions per second, and a memory of 2 megabytes; more than sufficient for the needs of SRFCS.

The standard F-15 (H009) data bus communicates with the inertial measuring unit, the attitude and heading reference set, a horizontal situation indicator, an air data computer, a central computer and a cockpit navigation control indicator. The control indicator is used for pilot communication and control of the test flight systems.

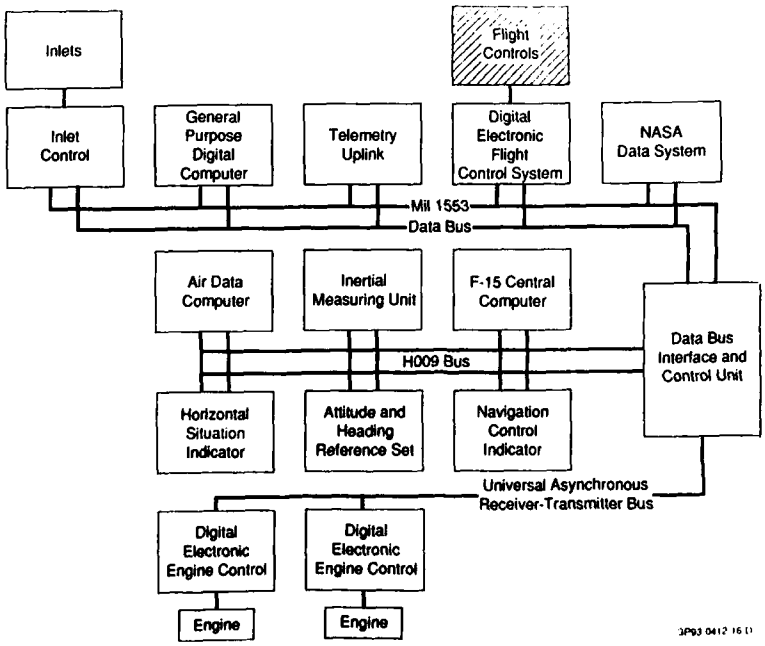



**Figure 17. Avionics Systems of F-15 Test Aircraft**

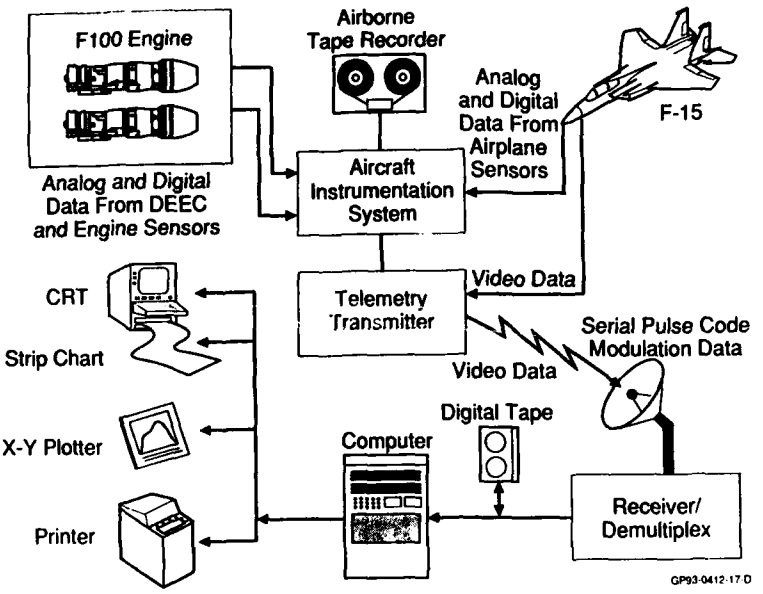



**Figure 18. The NASA Flight Data Acquisition System Provide Real Time Flight Data**

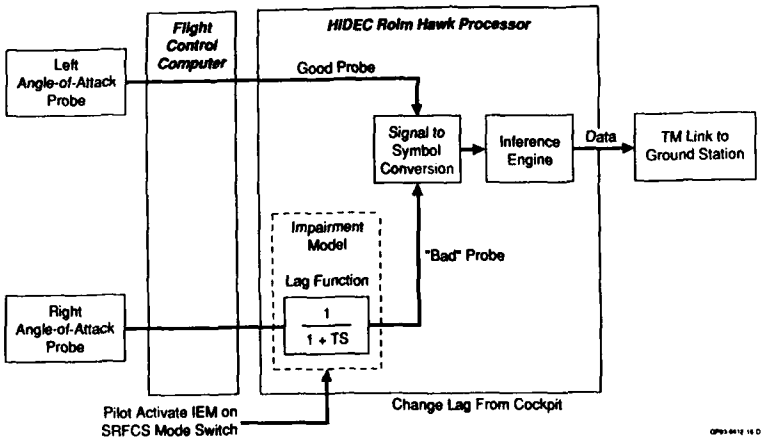



**Figure 19. Initial SRFCS Flight Assessment**
Angle-of-Attack Sensor Fault Diagnostics

All of the avionics are integrated with the data system, control systems computers (flight and propulsion) and the general purpose digital computer through the data bus interface and control unit allowing a high degree of integration.

The NASA F-15 is equipped with a data acquisition system capable of measuring, recording, and telemetering over 1000 parameters, as shown in Figure 18. Digital pulse code modulation recording is available for low and moderate frequency parameters (up to 1000 samples/sec) and frequency modulation recording for high frequency parameters up to 20,000 Hz. Digital data from the military standard 1553 data bus and the universal asynchronous receiver-transmitter bus are also recorded.

An air data boom senses total and static pressure, and vanes that measure angle of attack and angle of side-slip provide air data.

Typical three axis attitudes, rates, and accelerations are measured. In addition, the inertial measuring unit provides typical inertial data. Pilot commands and surface positions are also measured.

All of the telemetered data from the F-15 airplane are available for real-time computation and display, using the system shown in Figure 18. The telemetry data is recorded and sent to a large real-time processing computer. Two control rooms can each display 96 channels of calibrated strip chart data and almost unlimited digital data on cathode ray tubes.

Flight Test Objective And Approach

Initial flight results have been obtained for the SRFCS program. These results were from the maintenance diagnostics system analyzing a failure to an angle of attack sensor.

The objective of the initial flight test was: (1) To obtain a preliminary assessment of the SRFCS concept, (2) Demonstrate and evaluate the SRFCS maintenance diagnostic performance with parametric variations of failure conditions, and (3) Demonstrate airborne expert system performance early in the SRFCS program allowing impact of any lessons learned in the final design.

The system was implemented in software in the F-15 HIDEC onboard Rolm Hawk computer. This mechanization is shown in Figure 19.

The signals from the aircraft angle-of-attack (AOA) probes are sent via the flight control system to the Rolm Hawk processor. There the sensor impairment is emulated to occur on the right AOA probe.

The impairment is a lagged AOA signal due to excessive friction in the AOA sensor vane. This fault is typical of the type of failure that occurs during flight but cannot be easily identified during ground tests because actual flight conditions are not duplicated. The friction lag value could be changed in the test configuration to determine the degree of failure that can be detected using the Expert System process.

On board the aircraft, both the good probe and the failed, or "bad", probe signals were sent to the signal to symbol conversion module in the Rolm Hawk processor.

A total of 19 combinations of conditions were tested. These are summarized in Figure 20. The operating mode described as "auto" allows data concerning the states of the system at the time of the CAS disengage to be stored. Then, the data was used automatically as needed during the inferencing process.

The "on-request" operation mode requires some additional action by the pilot - either some stick movement via maneuvering flight, or intentional stick movement such as stick raps as requested by the expert system to allow completion of the inferencing process.

The mode switch allows the system to operate in a "normal" mode with no emulated failures present or in the "IEM" mode which allows an impairment emulation mode (IEM) to be introduced, (lagged probe). Note that the AOA monitor, avionic status panel indicator and CAS are all simulated in the IEM software.

There were three thresholds that could be controllable during the flight test if desired. (1) Disengage threshold - When the absolute value of the left AOA probe value minus the right AOA probe value exceeds the disengage threshold. The AOA monitors trips, the avionics status panel indicator latches, and the CAS disengage. This value was set as shown in Figure 21 for the flight test. (2) Rate threshold - When the AOA probe moves at a rate whose absolute value is less than the rate threshold the probe rate is considered zero. When both AOA probes move at rates whose absolute values are less than the rate threshold, the probes are considered to be in a quiescent state. The normal value of the rate threshold is 6.0 degrees/sec. (3) Mismatch threshold - When the absolute value of the left AOA probe minus the right angle of AOA probe exceeds the mismatch threshold value the software declares one probe lagging the other. This was set as shown in Figure 21.

| Test Point | Operation Mode | Mode Switch | Time Constant | Thresholds | | |
|---|---|---|---|---|---|---|
| | | | | Disengage (deg) | Rate (deg/sec) | Mismatch (deg) |
| 1 | Auto | IEM | 0.50 | 2.0 | 3.0 | 0.3 |
| 2 | Auto | IEM | 0.25 | 2.0 | 3.0 | 0.3 |
| 3 | Auto | IEM | 1.00 | 2.0 | 3.0 | 0.3 |
| 4 | Auto | Normal | 0 | 2.0 | 3.0 | 0.3 |
| 4A | Auto | Normal | 0 | 2.0 | 3.0 | 0.3 |
| 5 | On Request | IEM | 0.50 | 2.0 | 3.0 | 0.3 |
| 6 | On Request | IEM | 0.25 | 2.0 | 3.0 | 0.3 |
| 7 | On Request | IEM | 1.00 | 2.0 | 3.0 | 0.3 |
| 8 | On Request | IEM | 0 | 2.0 | 3.0 | 0.3 |
| 9 | On Request | Normal | 0.50 | 2.0 | 3.0 | 1.0 |
| 10 | Auto | IEM | 0.50 | 2.0 | 6.0 | 0.3 |
| 11 | On Request | IEM | 0.50 | 2.0 | 6.0 | 0.3 |
| 12 | Auto | IEM | 0.50 | 3.0 | 3.0 | 0.3 |
| 13 | Auto | IEM | 1.43 | 2.0 | 3.0 | 0.3 |
| 14 | On Request | IEM | 1.43 | 2.0 | 3.0 | 0.3 |
| 15 | Auto | IEM | 0.50 | 3.0 | 6.0 | 0.3 |
| 16 | On Request | IEM | 0.50 | 2.0 | 6.0 | 1.0 |
| 17 | On Request | IEM | 0.50 | 3.0 | 3.0 | 0.3 |
| 18 | On Request | IEM | 0.50 | 3.0 | 6.0 | 0.3 |
| 19 | Auto (See Note 1) | Normal | 0 | 2.0 | 3.0 | 0.3 |

Note 1: A special software program was used to invert the right AOA signal by 180° – a condition that is designed not to be considered lagged by the logic.

GP93 0412 19 D

**Figure 20. Initial SRFCS Flight Evaluation - Operational Modes**

In tests 1, 2 and 3, the system time constant was varied from 0.25 to 1.0 seconds while the operation mode was automatic and the mode switch was "IEM."

The right AOA mechanical system was always identified correctly as the failure mode. The time from CAS disengage until the identification of the failed unit was 1.5 seconds in each case.

For test points 4 and 4a, there was not lag applied. The system operated normally and no CAS disengage occurred with a pilot commanded input or in the presence of wake turbulence of the case aircraft with the distance between the F-15 from the chase aircraft gradually reduced from 3000 ft to 200 ft.

For test points 5, 6 and 7, in the "on-request" mode, with probe failure model lags from 0.25 to 1.0 seconds, the lag was not sufficient to trigger a CAS disengage. Note: the stick input was slightly less than in test part 2; therefore, 0.25 seconds is slightly below the limit of acceptable lags for testing purposes. The 0.5 second lag was sufficient to allow the CAS disengage when the pilot maneuvered, but, with the "on-request" mode, the pilot was required to perform a stick rap to complete the inferencing process and identify the right AOA mechanical system as the system that failed. Normal maneuvering during flight was not sufficient and the stick rap was requested 8 seconds after the CAS disengage. The total failure identification time was 9 seconds.

With a 1 second time constant on the lag, the normal pilot maneuvering was sufficient to complete inferencing 4 seconds after CAS disengage. Test point 8 was "on-request" with no lag applied to the right AOA signal and no CAS disengage occurred. Test point 9 has a CAS disengage occur even with an increased mismatch threshold and required a stick rap to complete the failure identification. The total time from CAS disengage to failure identification was 64 seconds.

Test point 10 increases the rate threshold but still disengaged CAS during maneuvering flight. Time from CAS disengage to failure identification was 1.5 seconds as was typical for the auto mode.

Test point 11 was the same as 10 except in the "on-request" mode and required stick raps to complete the failure identification. The stick rap was requested 11 seconds after CAS disengage. The time from CAS disengage until failure identification was 12 seconds.

Test points 12 and 13 were with lags of 0.5 and 1.43 seconds respectively with a disengage threshold of 3 degrees per second for test point 12 and 2 degrees per second for test point 13. CAS disengage occurred on both conditions during maneuvering flight with a time of 1.5 seconds from CAS disengage until failure identification.

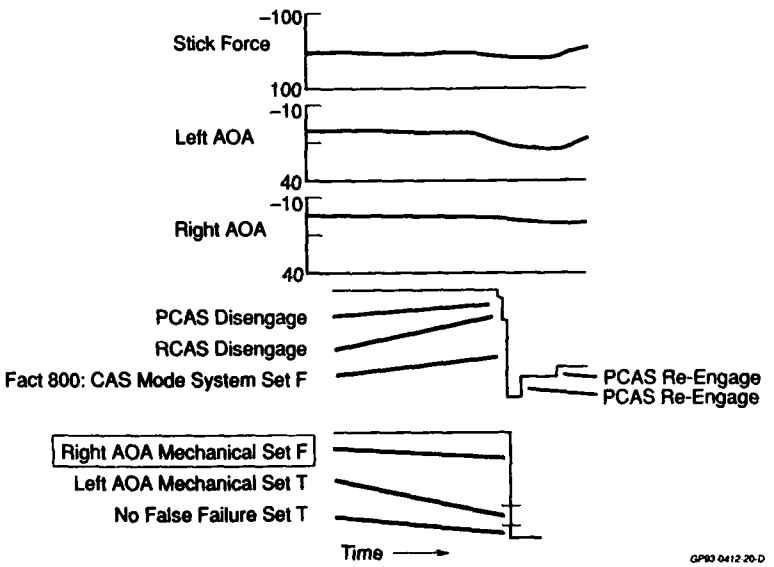



**Figure 21. Flight Recorded Diagnostic Results**
Failure to Right Angle-of-Attack Sensor

Test point 14 had the 1.43 second time constant but went to the "on-request" mode. CAS disengage occurred but no stick rap was required to complete the inferencing which required 3 seconds from CAS disengage until the failure was identified.

Test point 15, with the disengage threshold set to 3.0 degrees/second, a time constant of 0.5 and a rate threshold set to 6.0 degrees/second in the auto mode, had the right AOA mechanical system identified 1.5 seconds after CAS disengage occurred.

Test point 16 went to "on-request" mode and set the disengage threshold at 2.0 the rate threshold to 6 degrees per second, and the mismatch threshold to 1.0 degrees. The CAS disengage occurred but a stick rap was required to complete the right AOA mechanical system failure identification with a total time of 11 seconds.

Test point 17 set the disengage threshold to 3.0 degrees, the rate threshold to 3.0 degrees per second, the mismatch threshold to 0.3 degrees, and the lag time constant was set to 0.5 seconds. Two maneuvers were required to obtain CAS disengage. A stick rap was then required to complete the right AOA mechanical system failure identification. Stick rap occurred 10 seconds after CAS disengage time from CAS disengage to failure identification was 11 seconds.

Test point 18 set the rate threshold to 6.0 degrees/second with similar results to test point 17.

Test point 19 was a special case where the right AOA signal is inverted, i.e. 180# out of phase, in order to force the two probes apart such as a sharp wind disturbance would cause. (The logic is designed not to identify this condition as a probe failure). The system operated properly.

Example of Test Results:

Data from one test point will be evaluated in detail to show the type of pilot command and the typical signal responses. Examples of actual flight test data for the test point is shown in Figure 21. The data plot of stick force vs. time shows stick force increasing to 20 lbs. The left and right angle-of-attack probe signals start increasing for both probes responding to the increased stick force. The left probe signal responds faster than the right probe signal. In this example the right probe signal has a first order lag with a time constant of one second.

The delay in response can easily be seen from observing the difference between left and right AOA. The error between the left and right angle-of-attack probe signal causes a CAS disengage as shown on the discreet fact activation plots. The trigger fact is set indicating the CAS disengage. Additional facts (not shown) are set with the resulting conclusion that right AOA mechanical system is set false which indicates the right angle-of-attack probe is bad. This can be seen as the fact "Right AOA mechanical Set F", indicating that the correct answer has been inferenced by the Expert System diagnostics.

Summary of Flight Test Results

The flight test results indicate that in every case when the CAS disengaged the right angle of attack mechanical system was correctly identified as failed for both AUTO and "On-Request" modes.

The "on-request" mode generally required a stick rap to complete the inferencing process when the lag was less than 1.0 seconds. In these cases inferencing was always completed immediately following the rap.

For test program, either Auto or "on-request" mode could be implemented. In a production system that used this software, the automatic mode would be preferable to the "on-request" mode. Most important, the system correctly identified "No Fault" when the probes mismatched due to aerodynamic flow effects, thus eliminating unnecessary ground troubleshooting action.

In summary, the maintenance diagnostic system correctly identified the failure and isolated the problem. No false alarms was obtained even through the system was flown for many hours in the no lag state to evaluate its robustness to normal variations in a flight environment. This expert system approach to a complex dynamic fault monitoring problem aptly illustrates the potential of intelligent systems to reduce maintainability cost.

## 8.0 FUTURE SRFCS DEVELOPMENT PROGRAMS

The SRFCS program office will soon begin an Advanced SRFCS flight test program. Advanced SRFCS will build on the knowledge gained from the F-15 test program by expanding real-time reconfiguration and airborne maintenance diagnostics to cover more failure/damage modes over the test aircraft's entire flight envelope. The flight envelope will include landing tasks, supersonic flight, and flying with automatic modes, such as terrain following/terrain avoidance.

The SRFCS program office is also continuing the technology transition process for ground-based, expert system maintenance diagnostics. Work has begun to expand the field testing of FCMDS to other (F-16) operational squadrons. A statistical data base will be generated from the field testing to quantify the anticipated reduction in mean time to repair, and other metrics that describe maintenance efficiency.

The vision of the SRFCS program is full exploitation of SRFCS technologies in future aircraft designs. These aircraft will be less vulnerable than current aircraft to many types of battle damage. This means safe return of pilots and equipment following damage. These aircraft will have greater warfighting capability and they will be able to better complete tactical missions following damage or failures. The aircraft will have higher availability rates due to more efficient maintenance, and can better be maintained in austere environments. Weight reductions in the flight control system from reduction of redundant hardware provides improved performance and range. These aircraft will cost less over the lifetime of the fleet due to a less complex, more reliable flight control system. Reconfiguration algorithms improve performance of healthy aircraft by commanding control surfaces to optimize range or other performance parameters. Expert maintenance diagnostics connected to automated damage repair and dispatch planning systems increase aircraft availability. SRFCS technologies can be applied

to any digital control aircraft, including fighters, transports, close air support aircraft, special operations aircraft, and civil aircraft. The NASA F-15 flight test is an important step in transitioning the maturing SRFCS technologies towards the goal of increased mission capability and reduced life cycle cost.

## REFERENCES


1. "Control Reconfigurable Combat Aircraft, Architecture and Simulation Development," Final Report, AFWAL-TR-88-3118, Grumman Aircraft Systems.

2. Anderson, J.; Clark, C.; Madsen, P.; and Unfried, F., "USAF AFTI/F-16 Self-Repairing Flight Control System (SRFCS) Simulation," NAECON 1987, Dayton, Ohio.

3. "Failure Detection, Isolation, and Estimation for Aircraft Flight Control Systems Subjected to Actuator Failure and Surface Damage," Final Report, WRDC-TR-89-3058, Charles River Analytics Inc, May 1989.

4. Reconfiguration Strategies for Aircraft with Flight Control Systems Subjected to Actuator Failure/Damage, Final Report, WRDC-TR-89-3052, Lear Astronics Corp, May 1989.

5. Weinstein, W.; Posingies, W.; Eslinger, R.; and Gross, H., "Control Reconfigurable Combat Aircraft Flight Control System Development," AIAA GNC 1986, Williamsburg VA.

6. Eslinger, R. A. and Chandler, P. R., "Self-Repairing Flight Control System Program Overview," NAECON 1988, Dayton, Ohio.

7. Stifel, J. M.; Dittmar, C. J.; and Zampi, M. J., "Self-Repairing Digital Flight Control System Study", Final Report for Period January 1980-
October 1987, AFWAL-TR-88-3007, General Electric Company, May 1988.

8. Parkinson, R.; Urnes, J.; Fifield, N.; and Schroeder, J. B.; "In-Flight Maintenance Diagnostics," NAECON 1989, Dayton Ohio.

9. "Flight Control Maintenance Diagnostic System (FCMDS)," R&D Evaluation Report, Technology Assessment. Contract F33615-85-C-3613, Honeywell Inc, Systems and Research Center.

10. Burcham, Frank W., Jr., Gary A. Trippensee, David F. Fisher, and Terrill W. Putnam, Summary of Results of NASA F-15 Flight Research Program. NASA TM-86811, 1986 or AIAA 86-9761, 1986.

# FLIGHT TESTING OF A REDUNDANT FBW/FBL HELICOPTER CONTROL SYSTEM

by
H. Becker, K. Bender, K.D. Holle, G. Mansfeld
DLR, Deutsche Forschungsanstalt für Luft- und Raumfahrt e.v
Institut für Flugführung
3300 Braunschweig, Germany

## SUMMARY

The DLR has designed and developed an experimental fault-tolerant four-axes flight control computer system for helicopters named DISCUS. The acronym stands for "Digital Self-healing Control for Upgraded Safety". The main objective for the design of this computer system was to get a tool for various research tasks related to fault-tolerance, control law design and flight testing of new technologies.

This paper describes the design features of the DISCUS flight control computer system, the hardware realization, the software functions implemented so far and results of flight tests, all this performed in close cooperation with German industry. Although the hardware and the executive software of the flight control computer system are designed for four-axes applications, in the reported phase the DISCUS system is first of all flight tested in the yaw-axis control mode only.

## ABBREVIATIONS

| | | | |
|---|---|---|---|
| ACT | Actuator | FDC | Flight Director Computer |
| ADC | Air Data Computer | GEN | Electrical Power Generator |
| ADI | Attitude Director Indicator | HOL | High-order Language |
| AEU | Actuator Electronics Unit | HSI | Horizontal Situation Indicator |
| AFCS | Automatic Flight Control System | HST | Helicopter Simulation Terminal |
| AHRS | Attitude and Heading Reference System | H/W | Hardware |
| ARINC | Aeronautical Radio Inc. | HY-PWR | Hydraulic Power |
| ATS | Avionic Test Support | IFM | In Flight Monitoring |
| BIT | Built In Test | I/O | Input and Output |
| Coll. | Collective | LAT | Liebherr Aero Technik |
| CRT | Cathode Ray Tube | MBB | Messerschmidt Bölkow Blohm |
| DISCUS | Digital Self-healing Control for Upgraded Safety | NAV | Navigation Display (HSI) |
| | | PFD | Primary Flight Display (ADI) |
| EEPROM | Electrically Erasable Programmable Read-Only Memory | PFC | Pre-Flight Check |
| | | RAM | Random Access Memory |
| EFIS | Electronic Flight Instrumentation System | SBC | Single Board Computer |
| EPROM | Erasable Programmable Read Only Memory | SUB | Start-Up Built In Test |
| EPSU | Emergency Power Supply Unit | S/W | Software |
| FbL | Fly-by-Light | TRU | Transformer Rectifier Unit |
| FbW | Fly-by-Wire | ft. | feet |
| FCCS | Flight Control Computer System | kts. | knots |
| | | sec. | seconds |

## 1. INTRODUCTION

In order to improve handling qualities to ease the operation of helicopters, especially in bad weather conditions, the pilot must in future be supported in all flight phases by automatic control systems. This is accomplished by command control and autopilot systems with modes for specific missions, using high-precision navigation aids.

These requirements of operational performance of a helicopter will in future demand integrated digital flight guidance and control systems which have full authority on the control surfaces. These systems have to meet the same integrity requirements as the conventional hydro-mechanical systems used so far. In this context operational safety means fault tolerance. According to the present state of technology, the required operational safety can only be achieved by a redundant system with an appropriate redundancy management and voting/monitoring mechanism to identify and isolate any failure in all safety-related elements of the control system.

To investigate the problems related to fault-tolerance and redundancy management in safety critical flight control applications, especially for applications with helicopters, the DLR has launched a programme comprising the development and flight-testing of an experimental fault-tolerant flight control computer system (FCCS) using multiprocessor technology [1]. The system is designed as a four-axes flight-guidance/flight-control systems for helicopters.

The first operational application of this system was within a joint programme aiming at the development of a helicopter control system with optical data transmission, accomplished in close cooperation with MBB and LAT. This programme was sponsored by the German Ministry of Defense. Whilst MBB activities were aiming at a production-orientated yaw-axis control system [8], the DLR design of the fault-tolerant flight control computer system (FCCS) was laid out for a full four-axes application, although tested in this phase of the programme in the yaw-axis control mode only [7].

Both systems were flight-tested with the DLR test helicopter BO 105-S3 using the same sensor and actuator hardware and the same fibre optic data communication links.

Within this programme the prime contractor was MBB. The overall system design was accomplished by all partners, while the responsibility for component development was split between the partners:

- o MBB was responsible for the sensors, data communication, modification of the test helicopter and the production orientated dedicated yaw control system,
- o Liebherr-Aero-Technik (LAT) for the integrated electro hydraulic (smart) yaw actuator.

The four-axes flight control computer, the basic test helicopter equipment, flight testing, data acquisition, parameter determination of control laws and handling qualities evaluation was part of the DLR responsibility.

## 2. PROGRAMME OBJECTIVES AND REQUIREMENTS

The acronym used for the DLR project, carried out as part of the joint task is DISCUS, which stands for "Digital Self-healing Control for Upgraded Safety". The main objective for the design of the DLR flight control computer system was to get a tool for:

- o Research on and investigation of alternate methods to improve the integrity of flight control systems without increasing the redundancy level [2].
- o Development of advanced control laws to improve the handling qualities and thus to reduce pilot's workload.
- o Investigation and testing of new hardware technologies and components for FbW/FbL flight control systems (sensors, data communication, multi-processing, actuators).

With a tail rotor control system safety requirements may be met by a one-fail-op/fail-safe characteristic of the control system. On the other hand safety and mission requirements of an operational main rotor FbW/FbL control system require a two-fail-op capability and a total-loss probability of less than $10^{-9}$. At the present state of technology the above-mentioned requirements are met, for instance, by quadruplex, tri-duplex or dual-triplex configurations respectively. Along with the hardware-related reliability aspects in the design of a digital control system the possibility of dormant software errors and generic faults has to be considered. These errors occur on special operating conditions of the system not discovered by the performed test procedures. The general problem with software is that testing does not prove the absence of errors.

DISCUS is designed as an experimental system, and consequently does not necessarily need to have a two-fail-op capability as well as not to cover the problem of generic faults by dissimilarity. The DISCUS system therefore utilizes similar hardware and software in the redundant computers.

To ensure system safety and to meet the requirements for certification this implies that any failure in safety-critical parts of the experimental flight control system must not lead to a loss of the helicopter. Therefore the DLR FbW/FbL test helicopter BO 105-S3 is equipped with two independent control systems: the experimental FbW/FbL control system, operated by the test pilot and the mechanical backup system, operated by the safety pilot. In case of any malfunction (first failure) in the experimental flight control system the safety pilot has to take over control of the aircraft. This philosophy of performing flight tests provides the flexibility and a great potential for flight testing of new technologies at reasonable costs and manpower.

In addition to these safety provisions extensive tests in a ground testing environment have to be carried out before the flight testing in order to validate the experimental hardware and software. For this purpose a versatile ground-testing facility comprising the simulation of the helicopter's flight dynamics, the sensor system and the sensor interfaces has been set up.

The key element of the DISCUS project is the fault-tolerant flight control computer system (DISCUS-FCCS). The computer is designed as a modular experimental system according to the following major design requirements:

- o One-fail-operational capability (for flight testing mechanical backup is mandatory)
- o Low probability of total loss
- o Modular hardware design with adaptability to future research projects
- o Compatibility to a commercial standard
- o High computing power, capability of multi-processing
- o Extensive use of a high-order language (HOL); only low-level interface drivers written in assembly code

The software design and development of the DISCUS system was supported by means of software development tools.

## 3. SYSTEM DESCRIPTION

### 3.1 System Design and Architecture

Substantial reasons for the definition of the DISCUS system architecture were the method of monitoring, the desired one-fail-op capability, the feasibility to perform the planned research tasks, hardware flexibility and modularity and the existing servo actuators from a previous programme. Figure 1 shows the general architecture which both the yaw-axis as well as the future four-axes FbL control system have in common. Each lane has a dedicated set of sensors. Cross-strapping of the inputs is not applied, as it increases the interfacing effort without a substantial increase of integrity compared with cross-channel communication. Unfortunately a discontinuity of the redundancy structure was inevitable, as the design of the actuator was derived from an earlier programme, without significant changes to the hydro-mechanical part of the actuator. The required one-fail-op capability is achieved by a dual-duplex actuator and triplication of sensors, data links, flight control computers and electrical power supplies.

The DISCUS computer (FCCS) is designed for the four-axes FbW/FbL control system of the BO 105-S3 test helicopter of the DLR [3]. All the interfacing and computing power of the FCCS is already prepared for the four-axes system. In the first phase of the programme the DISCUS flight control computer system is flight tested in the yaw-axis control mode. The integration of the FCCS into the full authority FbL yaw-axis control system is shown schematically in Figure 2.

The general concept of the yaw-axis control system was governed by the following aspects [8]:

- Minimization of yaw response due to collective inputs employing an appropriate feedforward function
- Feedback loop gains in order to achieve robustness against parameter variations
- Simple system structure with a minimum effort in flight state sensors:
  - triplex yaw rate gyro as single flight state sensor
  - triplex pedal transducers
  - triplex collective transducers
  - triplex pilot's control and fault indication unit
  - triplex flight control computer
  - dual-duplex smart actuator
- Data transmission via optical fibres

The DISCUS FCCS consists of three electrically isolated parallel channels or lanes (Figure 3). The lanes are distinguished by the letters A, B and C. Each lane comprises three Single Board Computers (SBC) which perform different portions of the flight software tasks. Data to and from the various external systems (pilot controls, sensors and actuators) are transferred via interface cards and the standard VME-Bus to the SBCs. To identify each SBC in each lane, the SBCs are distinguished by the numerals 1, 2 and 3. Thus the first SBC in Lane A is called "A1".

Each SBC consists of a Motorola MC 68020 microprocessor with a MC 68881 math co-processor, 128 k-bytes Random Access Memory (RAM), 64 k-bytes of EEPROM and 64 k-bytes EPROM for the monitor and Start-Up BIT (SUB) software, two programmable timers and serial and discrete I/O. The monitor is used for programme loading, debugging and computer lane initialization on 'power-up'.

There are two means of communication between the SBCs in the redundant system:

- Within one lane, for Intralane Communication a 128 k-byte Global Memory is provided to exchange and share data between the SBCs.
- Between the three lanes, the data exchange is provided by a conveniently named Interlane Communications card.

The Intralane data transfer between SBCs, Global Memory and all interfaces is accomplished by a 16-bit parallel VME-Bus. All boards are plug-compatible to the commercial VME-Bus standard, but shorter in height to fit into ARINC boxes. Within each lane each SBC has equal rights to read data from or write data to the Global Memory. This may be compared with a COMMON block in FORTRAN but accessed by concurrent tasks. To avoid collisions during access, a round robin bus arbitration logic is used. Furthermore provisions have been taken by means of software in order to prevent one SBC from reading half-written data from another.

The specifically designed Interlane Communication card drives the fibre optic links from each lane to the other two lanes. Transmission is of serial type, utilizing a protocol similar to the ARINC 429 standard. Data are broadcast from one lane to the others. For instance, Lane A will send data to Lane B and C simultaneously. Each Interlane card has a 1 k-word mailbox memory for the incoming data received from the other lanes. Each of the three SBCs in one lane is capable to receive data via these memories or to send data to the other lanes.

As the DISCUS FCCS is designed for general purpose applications, the computer provides a set of standard and nonstandard interface cards for optical, analogue, discrete, ARINC-429 and MIL-1553 inputs and outputs. For the yaw control application the fibre-optical interfaces are used for data communication with sensors and actuators, while for four-axes activities most of the other interfaces are required. The DISCUS computer system was jointly designed by DLR and LAT (Liebherr Aero Technik). The hardware for both ground and aircraft installation was assembled by LAT. Figure 3 details one lane of the flight computer.

### 3.2 Software

In each lane of the triplex system identical software is loaded. The Fly-by-Wire executive is the top-level software which controls the entire system. Lane-dependent modules are selected from software by checking hard-wired lane-specific discrete inputs. Maximum performance is achieved by separating the software tasks between the three SBCs and operating them in parallel at appropriate cycle rates.

In the basic mode the pilot's command inputs are consolidated, written to the command output and, before being sent to the actuator, are consolidated once again. If the consolidation process detects any mismatch, it is recorded in the Error Matrix. Application-dependent flight control software is embedded into this basic executive and selected via the Control Unit by the pilot. Figure 4 shows the data flow of the basic mode of the executive.

The DLR system presently operated in the yaw-axis control mode already uses the executive of the future four-axes control system.

### 3.2.1 Synchronization

The three lanes of the DISCUS computer system are frame-synchronized to accomplish cross-channel monitoring of all inputs. To synchronize the parallel lanes data transmission by the Interlane card is performed, while Intralane synchronization uses the Global Memory.

Each SBC synchronizes with all SBCs in its own lane as well as with those in the other lanes. Thus a single fault in the synchronization of one SBC does not result in a total loss of one lane but just the loss of this specific SBC. All the other SBCs perform their normal tasks. As the individual SBCs in one lane operate at different frame intervals, the slowest frame rate determines the synchronization update rate.

### 3.2.2 Cross-Channel Monitoring (Voting/ Monitoring function)

For fault detection and isolation the three individual lanes have to exchange their data to perform cross-channel comparison. This is accomplished, too, via the Interlane Communications card. In each lane its own data are compared with those received from Interlane.

The monitoring of the data is accomplished by two different methods:

- o The first method is the cross-channel comparison with threshold values. This is used for all sensor signal inputs. The algorithm employed always selects the mid-value as the "true" one which is made identical for further use in all three lanes.
- o The second one is a cross-channel comparison with bit-by-bit voting. This is applied to those data which have been derived from inputs consolidated by the first method before. The "true" value is selected by majority voting. This method is applied to the integrators and the output signals.

Applying these two methods for cross-channel comparison reduces the difficult task of defining thresholds for the input signals only. It also simplifies the software function and increases computing efficiency. In a frame-synchronized system the estimation of the threshold values has to consider the inherent time lag between the lanes and the rate of the input signals. Bit-by-bit voting requires that data of matching frame cycles are to be compared.

### 3.2.3 Fault Management

The results of all monitoring functions in the flight software are written into an Error Matrix in the Global Memory. The matrix is divided into four different areas, associated with errors in the flight control software, the Start-Up BIT (SUB), the Pre-flight Check (PFC) and the interface device driver routines. Both, SUB and PFC will be described below.

Detected and consolidated errors are indicated to the pilot by the Fault Indicator Lamps in the Control Unit. Each lane has its own lamp in this Fault Indicator. An error is detected if one of these lamps is on. While the SUB and PFC switch the lamps on in case of just one single error, the flight control software counts the number of occurrences of an individual failure. Only if a predefined number of consecutive identical faults is reached, the error is indicated.

Any indication of a fault in the experimental system requires switching back to the mechanical backup system.

### 3.2.4 Power-Up and Start-Up Built-in Test

After 'power-up' each computer performs a Start-Up BIT (SUB) [4], which is a lane self-check. Then the synchronization of all processors in all lanes and operation of the basic "Fly-by-Wire" mode starts, which is the four-axes 1:1 FbW/FbL control mode. It is planned for the near future to run a subset of the SUB as an In-Flight-Monitor (IFM) during the idle time in each frame cycle, i.e. the time not consumed by the executive and the flight control software.

The SUB performs hardware and software tests of the SBCs, the global and local memories, the timers, the interfaces and the power supply. In the case of a temporary power loss of one lane during flight, after power recovery the SUB is omitted to reduce the time of temporary loss of one lane.

### 3.2.5 Pre-flight Check

An overall system test before take-off is performed by the Pre-flight Check function (PFC). While the SUB is running asynchronously within each SBC, the PFC requires synchronization of the SBCs. The PFC is initiated by pressing a designated button on the Control Unit (Figure 2). The tasks of the PFC are devided between the SBCs within one lane to increase efficiency and to test the individual SBCs. Besides others one goal of the PFC is testing of fault indication and management by fault stimulation to ensure system safety and the absence of dormant errors before take-off. Presently the following tasks are implemented:

- o Command Input and Actuator Test
- o Test of the Cross-Channel Comparison
- o Synchronization Test
- o Fault Indication Test

Additionally the fault analysis is checked by cross-channel comparison of the resulting Error Matrices. Furthermore, for maintenance reasons a fault history is generated.

Both, SUB and PFC, are also part of the verification and validation programme of the DISCUS software and hardware.

## 4. GROUND TESTING ENVIRONMENT

To validate the flight software a ground testing environment has been set up. This comprises all the helicopter system interfaces with the defined redundancy level, a simplified non-linear flight simulation of helicopter dynamics, a simulation of sensor and hardware errors, interfaces for general-purpose test equipment and data links to a helicopter mock-up. Additionally, genuine hardware as sensors for pilot controls and actuators may be included in this environment.

A schematic representation of the ground test environment is shown in Figure 5. Excluding the data link to the helicopter mock-up, the essential components of the test equipment are:

- o Helicopter Simulation Terminal (HST) Computer with the associated Console Terminal (Test Control Panel)
- o Error Display Unit
- o Strip Chart Recorder
- o Control Unit (Mode Control Panel)
- o Pilot Controls Mock-up
- o Actuator Test Rig
- o Triplex Flight Control Computer System

### 4.1 Helicopter Simulation Terminal Computer

The Helicopter Simulation Terminal (HST) is a computer system which comprises all those sensor data with the required redundancy level which are necessary to perform ground testing of the flight control computer system. To meet the real-time requirements a simplified non-linear flight simulation of the BO 105-S3 test helicopter dynamics is running for control law evaluation. Multiplication of all generated single-channel signals to achieve the required redundancy is performed by a software function.

### 4.2 Fault Simulation

To prove correct system behaviour in the case of a fault, a simulation facility was integrated into the HST software to generate different kinds of faults. Presently there are three types of faults which may be initiated by command input from the Test Control Panel:

- o Sensor faults
- o Control Unit faults
- o Hardware faults in the redundant computer system

The generation of sensor and control unit faults is performed by superimposing failure signals or delaying the correct signals. In the case of original hardware being used in the test, this requires the signals have to be passed through the HST computer.

A limited number of hardware faults in the test specimen, the redundant computer system, may also be controlled via the HST computer. This relates to hardware elements not accessible by other means of fault injection. Among others, deviations of the synchronization timers, memory faults, infinite processor loops are software-simulated.

Apart from this the usual error injection by signal and power interruption generated manually by means of a patch panel is available, too.

### 4.3 Error Display Unit

As mentioned above, a detected error in the redundant computer system is indicated by switching on the Fault Indicator lamps contained in the Control Unit.

Additionally, information about the type and location of the error is taken from the error analysis, run in each lane of the computer system. The analysis routines output an error code which is transferred to the HST computer. There a textual and graphical representation of the analysis result is accomplished. For cost saving reasons the Error Display Unit is a standard graphic computer display. For this device neither redundancy nor real-time performance was required.

### 4.4 Hardware Integration Test Rig

As shown in Figure 5 additional hardware, as the position sensors for pilot commands and the yaw-axis actuator, may be integrated into the test rig. The signal line (electrical or optical) may be coupled directly to the test specimen or passed through the HST computer for fault injection.

## 5. TEST HELICOPTER INSTALLATION

In order to perform flight testing using experimental hardware the DLR has to introduce certain modifications to their test vehicles (planes and helicopters). For all modifications DLR has to apply for compliance and approval with the civil airworthiness regulations according to e.g. FAR Part 27. That means civil certification of airworthiness is mandatory for flight testing. Common practice with regard to certification of safety-critical computer systems in civil fixed-wing aircraft is a dissimilar design in software and hardware (e.g. redundancy of control surfaces). As the DISCUS-System consists of similar hardware and software a backup control system is required to achieve the certification.

Since 1982 the DLR has been operating the BO 105-S3 at its site in Braunschweig as its Fly-by-Wire test helicopter. In the early seventies the helicopter had been equipped by MBB with a non-redundant four-axes Fly-by-Wire control system [9]. This equipment comprises two independent command paths, one with a single-lane Fly-by-Wire control system, the other with access to the basic hydro-mechanical control system which is common to all commercial BO 105 helicopters. For all flight tests two pilots are required. One pilot, the test and evaluation pilot, is responsible for every experimental system integrated into the helicopter. The second pilot is the safety pilot controlling the mechanical backup system.

The helicopter can be flown in two operation modes. In the normal FbW-OFF mode the safety pilot has full control over the helicopter via the basic hydro-mechanical controls. The second mode (FbW-ON) gives control with full authority to the test pilot while the safety pilot is able to take over via the backup system by overriding the actuators with a defined, limited control force without disengaging the system. This feature is provided by means of a preloaded spring in the actuator link, schematically shown in Figure 6. In addition to this feature the Fly-by-Wire system may be switched off (disengaged) electrically by both pilots.

The Fly-by-Wire system is engaged by synchronizing the control inputs of the test pilot with the basic hydro-mechanical controls of the safety pilot. This synchronizing feature is automatically performed by trim actuators on the test pilot's controls. Only in case the synchronizing process was successful, the Fly-by-Wire system may be engaged by pressurizing the servo valves of the yaw actuator and the test pilot takes over the command of the helicopter.

This system philosophy offers the capability of putting new advanced technologies into flight trials in very early stages of development. In the past a great deal of flight testing of new control laws and operating modes for helicopters has been performed very successfully with the BO 105-S3 single-lane FbW system [10]. This experience of control law evaluation by the DLR was brought into this programme.

Within this programme the single-lane Fly-by-Wire control system has been upgraded to a redundant fault-tolerant control system in the yaw-axis. The block diagram of the yaw control system may be taken from Figure 2, while the installation of all subsystems into the test helicopter BO 105-S3 is shown schematically in Figure 7. Further development to expand this system to a redundant four-axes Fly-by-Light control is planned.

Data acquisition is performed by the so-called ATS pallet (Avionic Test Support) connected via a triplex fibre optic link to the test specimen, the FCCS. Furthermore the ATS system provides reference data for all tests by means of an AHRS (Attitude Heading Reference System), an Air Data Computer (ADC) and a Doppler radar. Besides this a symbol generator and two multifunction CRT displays are provided which can be programmed according to the flight test requirements. Finally the ATS system performs the data acquisition of the test specimen and the reference system respectively for transmission to the ground telemetry station where the engineers monitor and control the flight test on-line. A block diagram of the complete test and data acquisition system including the ground facilities is shown in Figure 8.

## 6. FLIGHT TESTING

### 6.1 The Control Law of the Yaw-Axis System

The task of both defining the control loop structure and evaluating the set of control loop gains for first flight was performed by MBB. DLR carried out the flight testing and the optimization of the control parameters for the handling quality evaluation.

The elaboration of the yaw-axis control laws aimed at tight tracking of yaw rate and the minimization of the influence of wind and gust disturbances especially relating to both side and tail winds. Additionally it was requested to reduce the strong and troublesome influence of collective command inputs to the yaw movement, commnly known with all helicopters of this type.

For cost saving reasons the yaw control system has to utilize a minimum number of sensors. The pilot's pedals are used for input of a yaw rate control command. In order to achieve the intended improvements the yaw control system must possess full authority of actuator displacement and rate. These requirements imply that the yaw control system should be fault-tolerant with a continuous self-monitoring and recovery capability. A one-fail-op, fail-safe capability pertained to all system components is believed to be sufficient for this kind of application, as the tail rotor has an automatic centering capability by means of centrifugal weights.

Figure 9 depicts the general structure of the yaw-axis control. As can be seen, two operating modes are implemented:

- o AFCS-OFF for direct steering of the tail rotor (1:1 mode)
- o AFCS-ON

In the AFCS-OFF mode the tail rotor blade angle is actuated directly by the pilot's pedal commands utilizing the FCCS and the optical data transmission.

The yaw-axis controller is switched to the AFCS-ON mode via the FCCS Control Unit. In this mode the pilot commands a desired yaw rate via deflection of the pedals. The control functions within the FCCS compare this rate command to the actual helicopter yaw rate yielding an actuation output to the tail rotor. In order to avoid structural resonance excitation a notch filter is implemented. A second notch filter is used to protect the tail rotor drive shaft from torque oscillations.

### 6.2 Test Programme and Test Results

In order to evaluate the expected improvements of the yaw control system eight different manoeuvres were defined for a flight test programme:

- heading jumps at +/- 20° and +/- 45° at constant airspeed (60 kts) and altitude
- orbit (full left and right turns) at constant airspeed of 60 kts and 45° bank angle
- dolphin manoeuvre (altitude changes of +/- 100 ft within 40 sec) at constant airspeed of 60 kts
- cruise flight at 60 kts with constant side-slip of +/- 20° or - 20° respectively
- hover flight at 50 ft of altitude and constant heading
- sideward movement
- hover flight with heading jumps of +/- 45° at an altitude of 50 ft
- hover flight with heading and altitude changes.

During the flight test programme four military pilots and two DLR test pilots had to fly these manoeuvres both in the "direct control" (AFCS-OFF) and the "automatic flight control" (AFCS-ON) mode. After the flight tests every pilot had to fill in a questionnaire concerning his opinion upon the pilot's workload experienced during the manoeuvres mentioned above.

In order to make the manoeuvres reproducible the pilots had to follow the indications of the localizer and glide slope deviation indicator on the primary flight display (PFD). These indicators show the difference between a commanded value, generated by a function generator implemented in the ATS system, and the actual measured value.

For the jump heading manoeuvres the function generator produces a sequence of left and right heading changes of a preselectable amount of +/- 20° to +/- 45° respectively. The deviation from the commanded heading is indicated by the localizer deviation indicator on the PFD. In addition the heading command signal drives the heading select bug on the NAV (navigation) display in rose mode.

If the pilot keeps the heading difference within +/- 2.5° for more than 5 seconds the next heading command step follows.

For the dolphin manoeuvre the function generator produces a triangularly shaped altitude pattern, with altitude changes of +/- 100 ft within 40 seconds. The deviation from the commanded altitude is indicated by the glide slope deviation indicator on the PFD. This manoeuvre requires high collective inputs which influence the helicopters yaw movement strongly.

Figure 10 shows results of the flight tests for a manoeuvre "hovering with heading and altitude changes". The reduction in pilot's workload is obvious by comparing the necessary pedal inputs in both modes. Figure 11 shows results of a hover flight with crosswind (with gusts up to 38 kts). With the "automatic-control-mode" engaged, no pedal inputs are required to keep the heading almost constant.

## 7. CONCLUSIONS

The results of the flight tests proved the advantages offered by the yaw control system concerning the handling qualities.

The fail-op capability of the DISCUS FCCS was proven during 50 hours of flight testing of the FbL yaw-axis control system. Both single random failures (mainly due to deficiencies in the optical connectors at the first flights) as well as intentionally injected errors never lead to a total loss of control.

## 8. ACKNOWLEDGEMENTS

The authors would like to thank the staff members at MBB and LAT, the partners in this programme and all those colleagues at the Institute for Flight Guidance and the DLR Flight Test Division who have contributed to the success of this programme. Special thanks to Mr. G. Hähnlein and Mr. H. Leyendecker who contributed to the flight test instrumentation and to the control law evaluation.

## 9. BIBLIOGRAPHY

[ 1] Mansfeld, G.
Konzeptvorschlag für ein fehlertolerierendes Fly-by-wire Steuerungs-/ und Regelungssystem für die Bo 105-S3.
DFVLR-IB 112-84/01, 1984.

[ 2] Bender, K.; Edinger, Chr.
Zuverlässigkeiten an Mikroprozessor-DV-Strukturen für Steuerungs- und Regelungssysteme.
DFVLR-IB 112-84/28, Braunschweig 1984.

[ 3] Bender, K.
DISCUS - Ein fehlertolerierendes FBW / FBL - Rechnersystem
DFVLR-IB 112-88/19, Braunschweig 1988.

[ 4] Sudduth, R.
Start-up Built-in Test for the DISCUS Fault-Tolerant Fly-by-Wire Computer System
DFVLR-IB 112-88/25, Braunschweig 1988.

[ 5] Becker, H.
Pre-Flight Check für ein fehlertolerierendes FBW-DV-System.
DFVLR-IB 112-88/20, Braunschweig 1988.

[ 6] Carl, U.B.
Elektrische Primärsteuerung (Fly-by-Wire) - Flugsteuerungssysteme neuer Technologie -
Electrical Primary Flight Control System (Fly-by-Wire) - Flight Control Systems of new Technology -
BMFT Fördervorhaben LFL83607, MBB Forschungsbericht UT-122-88, Bremen, Juni 1988

[ 7] B. Formica (MBB), H. König (MBB), K. Bender (DLR), G. Mansfeld (DLR)
OPSTI - A Digital Optical Tail Rotor Control System
Proc. 14th European Rotorcraft Forum, pp.91-1/91-16, Sep. 20.-23., Milano, 1988

[ 8] König, H.; Stock, M.; Zeller, S.
OPSTI - An Optical Yaw Control System for High Performance Helicopters
Proc. 45th Annual Forum & Technology Display of the American Helicopter Society, May 22-24, Boston, 1989

[ 9] Attlfellner, S.; Rade, M.
BO 105 In-Flight Simulator for Flight Control and Guidance Systems
Proc. of the 1st European Rotorcraft Forum, Southhampton, 1975.

[10] Leyendecker, H.
The Model Inverse as an Element of a Manoeuvre Demand System for Helicopters
Proc. of the 12th European Rotorcraft Forum, 1986.

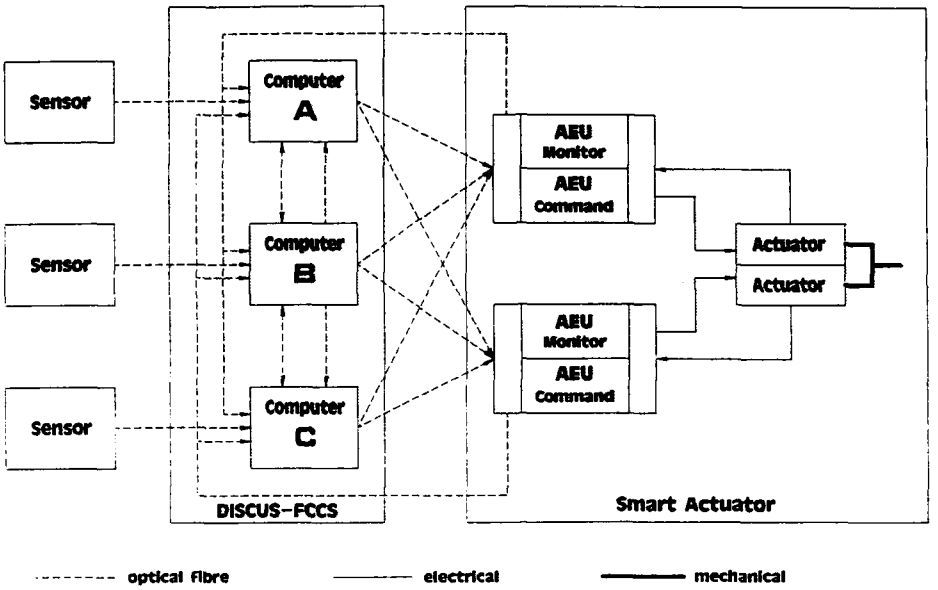



Figure 1 : Schematic Representation of DISCUS Redundancy Architecture

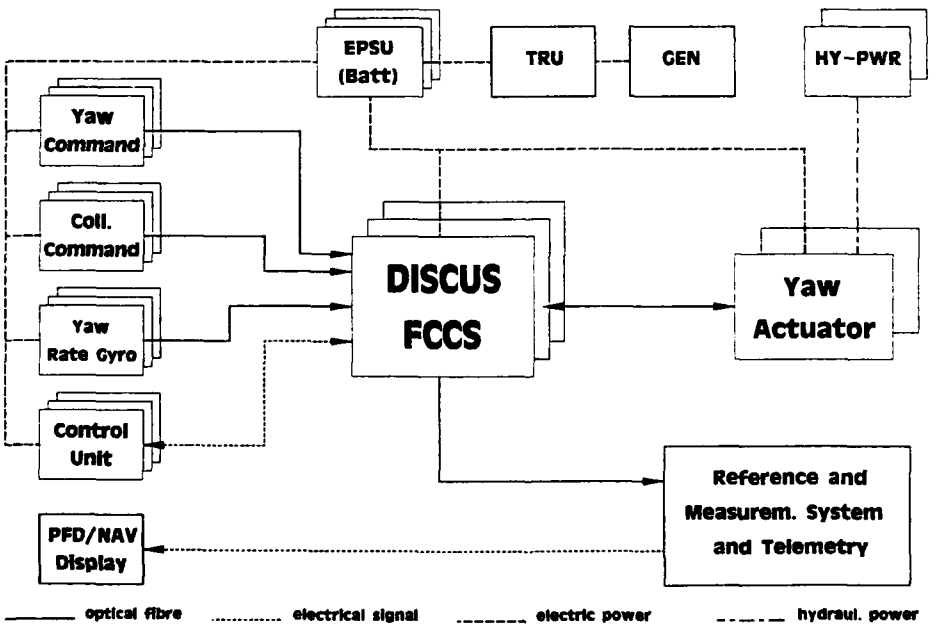



Figure 2 : Hardware Structure of the FbL Yaw-Axis Control System

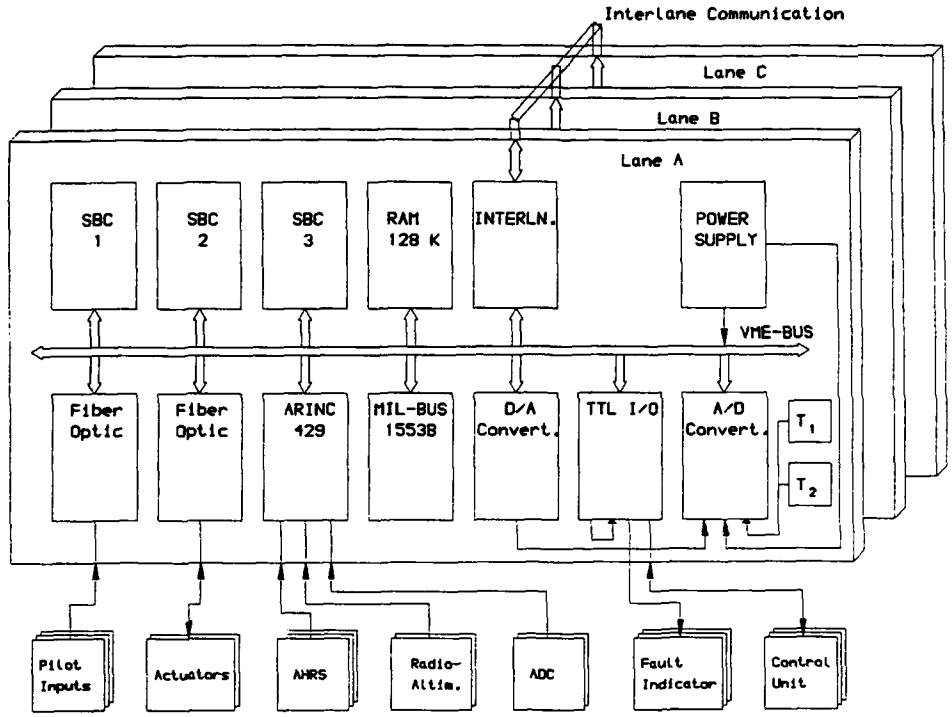



*Figure* 3 : One Lane of the DISCUS Computer Hardware

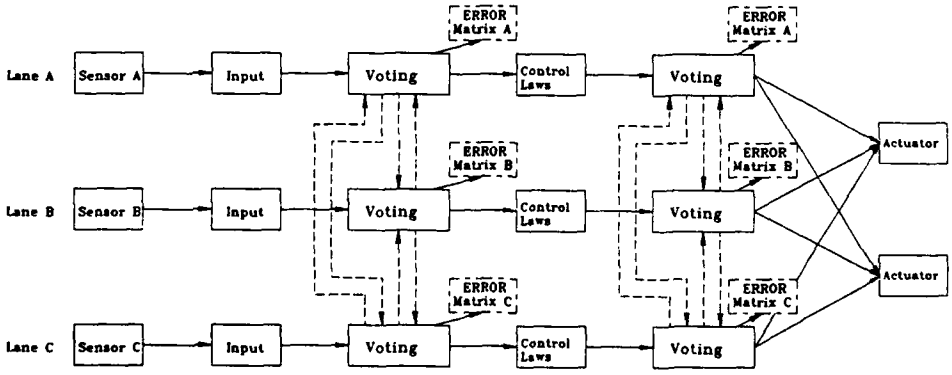



Figure 4 : Data Flow within the DISCUS System

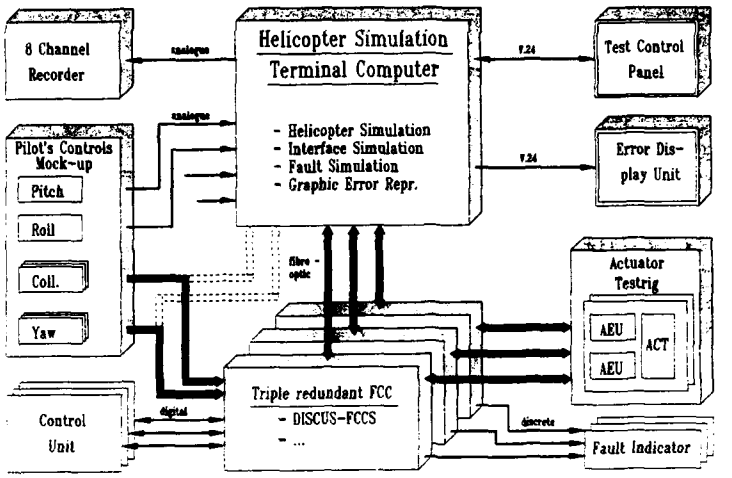



Figure 5 : Schematic Representation of Ground Test Environment

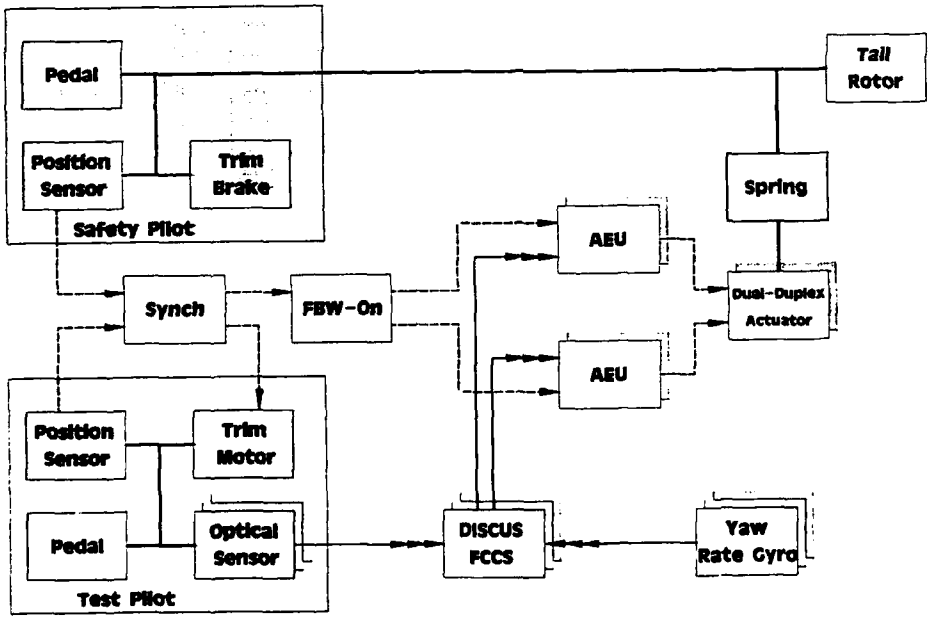



Figure 6 : Interconnections of Safety and Test Pilot's Controls on the BO 105-S3 Test Helicopter

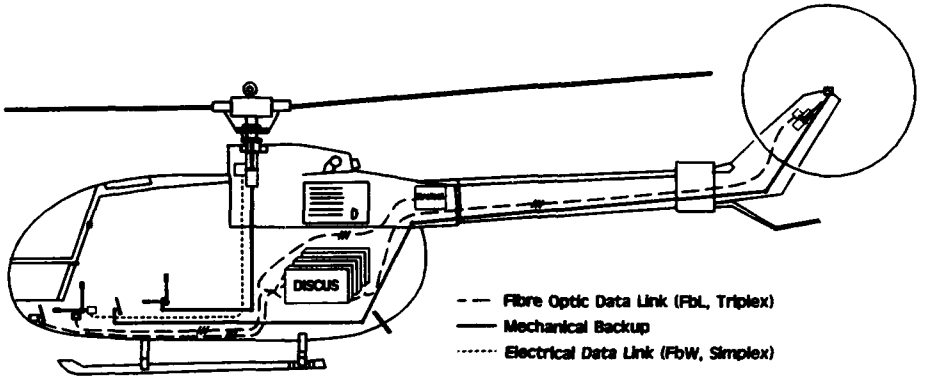



Figure 7 : Subsystem Installation into the BO 105-S3 Test Helicopter of the FbL Yaw-Axis Control System

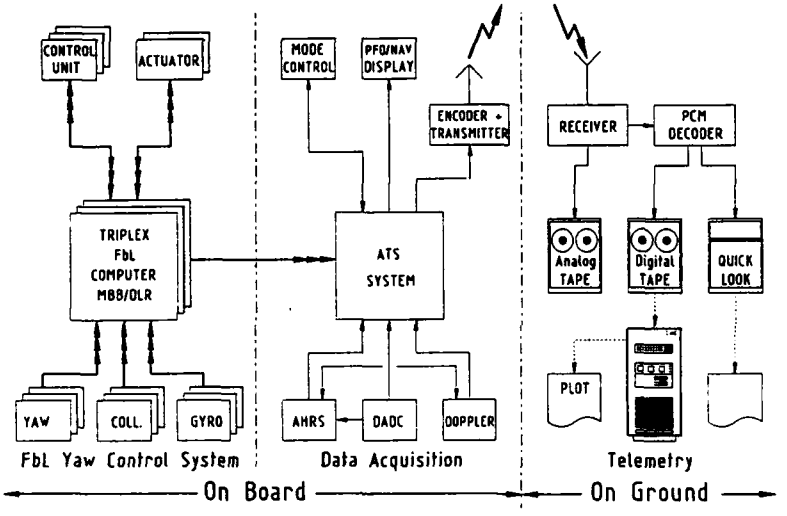



Figure 8 : Block diagram of the complete flight test system

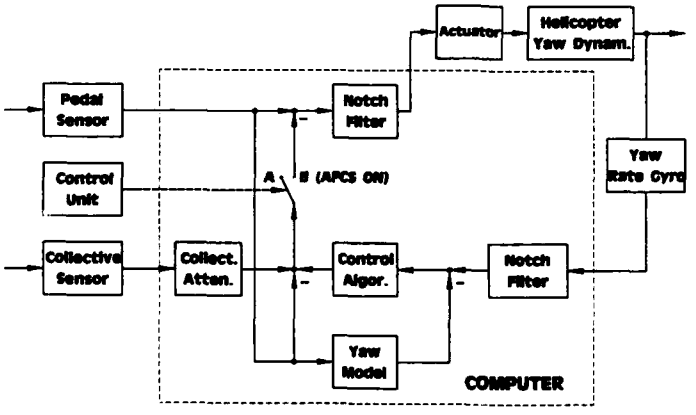
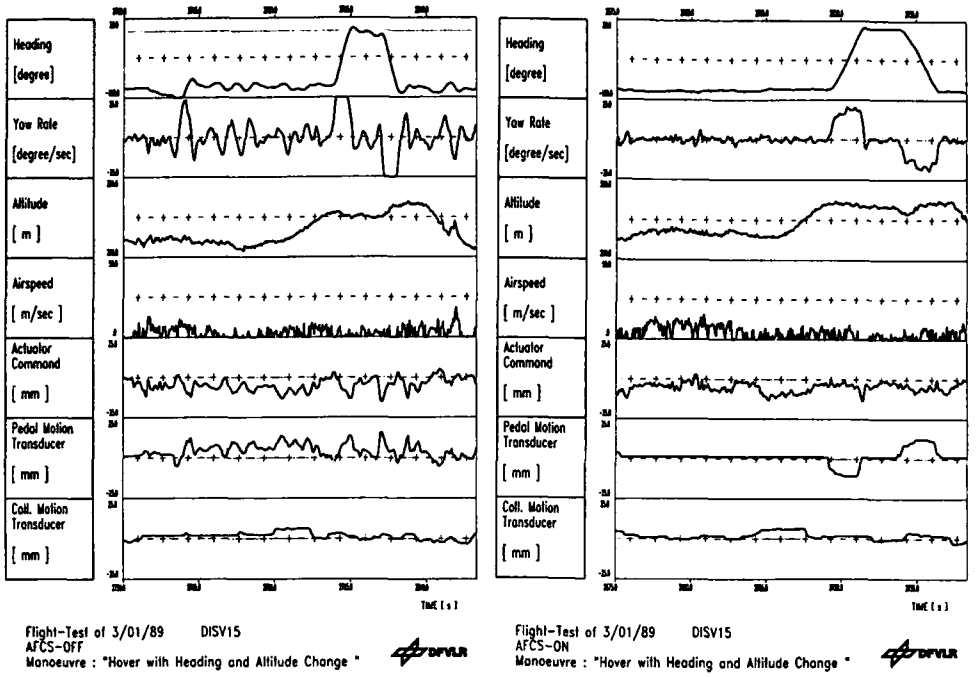


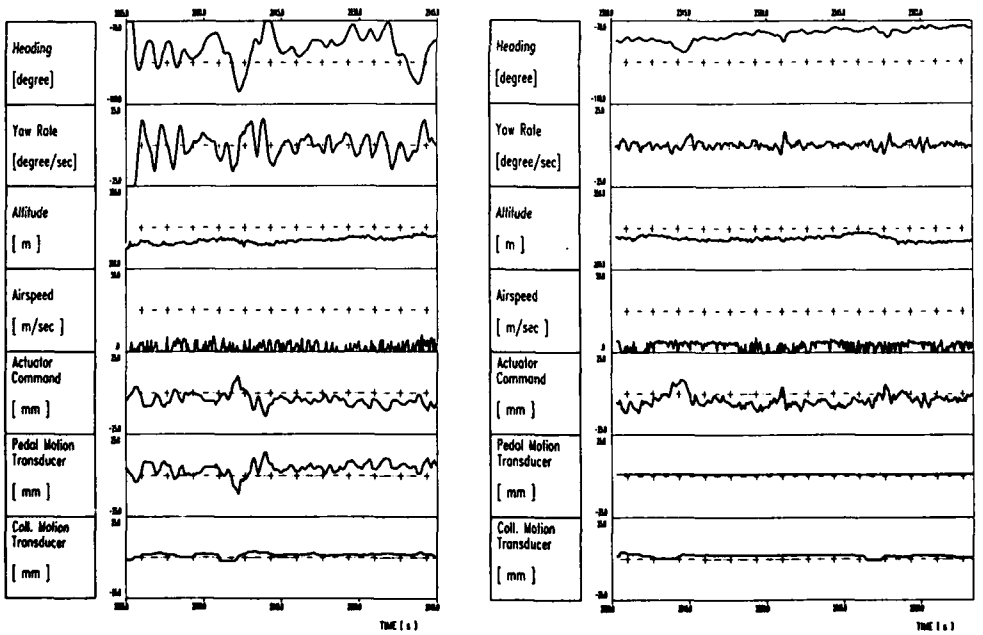

Figure 9 : Functional Block Diagram of the Yaw-Axis Control Law, Flight Tested with the DISCUS-FCCS




Flight-Test of 3/01/89 DISV15
AFCS-OFF
Manoeuvre : "Hover with Heading and Altitude Change "

Flight-Test of 3/01/89 DISV15
AFCS-ON
Manoeuvre : "Hover with Heading and Altitude Change "

Figure 10 : Test Results of the Manoeuvre "Hover with Heading and Altitude Change"




Flight-Test of 3/01/89 DISV15
AFCS-OFF
Manoeuvre : "Hovering with Crosswind from Left Side "

Flight-Test of 3/01/89 DISV15
AFCS-ON
Manoeuvre : "Hovering with Crosswind from Left Side "

Figure 11 : Test Results of the Manoeuvre "Hovering with Crosswind from Left Side"

# UN SYSTEME DE REFERENCES PRIMAIRES DE HAUTE INTEGRITE

par

JL. ROCH et J. CONTET

SEXTANT Avionique (CROUZET)
25, rue Jules Védrines

26027 VALENCE CEDEX - FRANCE

## 1. RESUME

Cet exposé décrit les solutions retenues pour la réalisation d'un Système de Références Primaires moderne destiné au pilotage et à la navigation d'avions et d'hélicoptères. Ce système critique pour le vol entre dans la chaîne de sécurité de l'avion et répond à un certain nombre de contraintes spécifiques. Dans un premier temps, la problématique qui impose ces contraintes est exposée, sont ensuite décrites les méthodes et les technologies permettant de les satisfaire.

## 2. INTRODUCTION

Le système présenté ici a été récemment développé par SEXTANT Avionique (CROUZET) pour constituer la source de références primaires nécessaires au pilotage et à la navigation dans le cadre des systèmes intégrés d'hélicoptères et d'avions civils ou militaires modernes. Il a notamment été retenu pour le programme d'hélicoptères SUPER PUMA MK2 de l'AEROSPATIALE, ainsi que pour le programme de rénovation des C160 TRANSALL de l'Armée de l'Air Française. Dans ces deux exemples, il est au coeur d'un système de conduite du vol ou de navigation hautement intégré, fourni par SEXTANT Avionique. Il fait également l'objet de nombreuses propositions en cours de sélection.

Sa fonction de base, tant pour les applications civiles que militaires, en fait un système critique, dont les exigences en matière de fiabilité, de sécurité, de tolérances aux pannes sont particulièrement sévères.

Enfin, par vocation, c'est un système susceptible de s'adapter à de nombreux porteurs, donc de satisfaire à des spécifications très variables d'environnement, d'interfaces, voire de modes de fonctionnement.

Le présent exposé décrit quelques unes des réponses qui ont été apportées à l'ensemble de ces contraintes, en particulier dans le cadre de l'application SUPER PUMA MK2.

## 3. PRESENTATION DU SYSTEME DE REFERENCES PRIMAIRES - PROBLEME POSE

### 3.1 Fonctionnalités

Dans le cadre du SUPER PUMA MK2, le SRP est un système de base qui a pour fonction de fournir aux divers systèmes de bord (visualisation, PA, navigation, etc...) les informations, dites de références, suivantes :

- Cap par rapport au Nord magnétique
- Attitudes
- Vitesses angulaires en axes porteur
- Forces spécifiques en axes porteur
- Vitesses air : vitesse indiquée, vitesse propre et vitesse verticale
- Altitude pression standard
- Température extérieure

D'autre part, le SRP est connecté à un radar Doppler afin de fournir une vitesse sol non bruitée, obtenue par filtrage complémentaire des vitesses Doppler avec les accéléromètres du SRP.

3.2 Architecture

Le SRP est composé des équipements suivants :

- Un FDC (Flight Data Computer), installé en soute, qui contient les capteurs inertiels et l'unité de traitement et d'interface du SRP.

  Cette unité, qui constitue le coeur du système, est la Centrale de Références Primaires CIRUS

- Un HSU (Heading Sensor Unit) qui est constitué par un magnétomètre CROUZET type 110 S3S

- Un PSU (Pressure Sensor Unit), installé en soute, qui comprend les capteurs de pression (CROUZET type UMP 300)

- Un TPU (Temperature Probe unit) qui est une sonde de température du type CROUZET 20-2 connecté au PSU

Le SRP est relié aux autres systèmes ou équipements de bord par des liaisons numériques du type ARINC 429, des discrets de commande et d'état, et quelques liaisons spécifiques.

Pour des raisons de fiabilité et de sécurité, l'ensemble du système est entièrement redondé.

La composition du système est décrite par la Figure 1 ci-après.

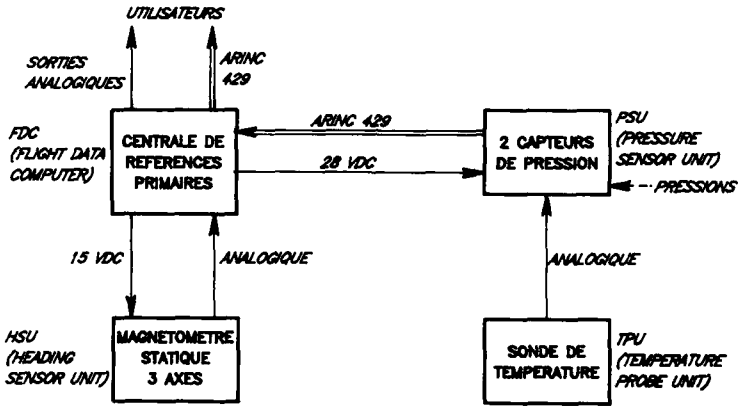



**FIGURE 1 : CONSTITUTION DU SYSTEME DE REFERENCES PRIMAIRES**

L'architecture générale du système intégré du SUPER PUMA MK2 est, quant à elle, décrite par la Figure 2. Une présentation plus complète du système du SUPER PUMA MK2 a déjà été faite (Ref. 1 et 2).

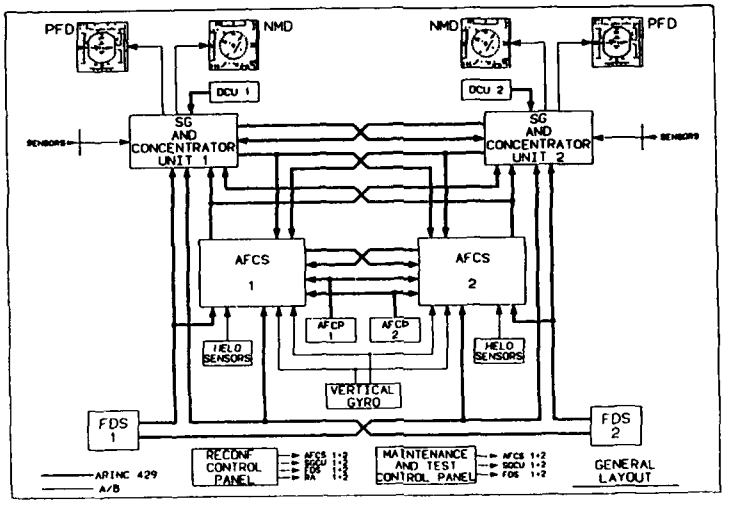



**FIGURE 2 : SYSTEME INTEGRE DE CONDUITE DU VOL DU SUPER PUMA MK2**

***(IFDS : Integrated Flight and Display System)***

3.3 Exigences *en matière de fiabilité et de sécurité*

L'ensemble du système doit être certifié Aviation Civile (DGAC, FAA, CAA), en tant que fonction critique sur l'hélicoptère. Il en résulte un certain nombre de contraintes, parmi lesquelles :

- Classification du logiciel complet en "niveau 1" au sens de la norme DO 178A
- Exigence de fiabilité élevée, soit un MTBF de plus de 2500 H de vol pour l'ensemble du système
- Exigences de sécurité élevées :

| Evénements redoutés | Exigences |
|---|---|
| Perte signalée des paramètres critiques ou essentiels : $\Psi$, $\theta$, $\varphi$, p, q, r, $\gamma y$ Vi, Vz, Zb, ts | $10^{-6}/h$ |
| Fourniture d'informations critiques erronées non signalées : $\Psi$, $\theta$, $\varphi$, p, q, r, Vi, Zb | $10^{-10}/h$ |
| Fourniture d'informations essentielles erronées non signalées : $\gamma y$, vz, ts | $10^{-8}/h$ |
| Fourniture d'informations non essentielles erronées non signalées : $\gamma x$, $\gamma z$, vitesses Doppler | $10^{-6}/h$ |

$\theta$, $\varphi$ : attitudes

$\Psi$ : cap

p, q, r : vitesses de rotation en axes machine

$\gamma x$ $\gamma y$ $\gamma z$ : accélérations en axes machines

Vi : vitesse indiquée

Vz : vitesse verticale barométrique

Zb : altitude standard barométrique

Ts : température statique

3.4 Evolutivité du système

Le système conçu pour les besoins de l'hélicoptère SUPER PUMA MK2 est également proposé sur ce nombreux autres porteurs, avions ou hélicoptères. Il a été ainsi retenu pour équiper le programme de rénovation TRANSALL C 160 de l'Armée de l'Air française.

Il en résulte une nouvelle contrainte : s'adapter aisément à ces systèmes très disparates :

- Au niveau des interfaces électriques. le plus souvent, les programmes de "retrofit" exigent simultanément la présence d'interfaces numériques, analogiques, synchros, discrets, en grand nombre.

- Au niveau des sources d'informations anémométriques : dans de nombreux cas, l'information est déjà présente sur l'avion, ce qui implique des modifications au niveau des entrées du système.

- Au niveau des contraintes d'environnement et des dynamiques de porteur, qui vont de l'avion de combat au transporteur civil, en passant par l'hélicoptère.

A titre d'exemple, la figure 3 montre la complexité et la diversité des interfaces nécessaires pour le programme TRANSALL

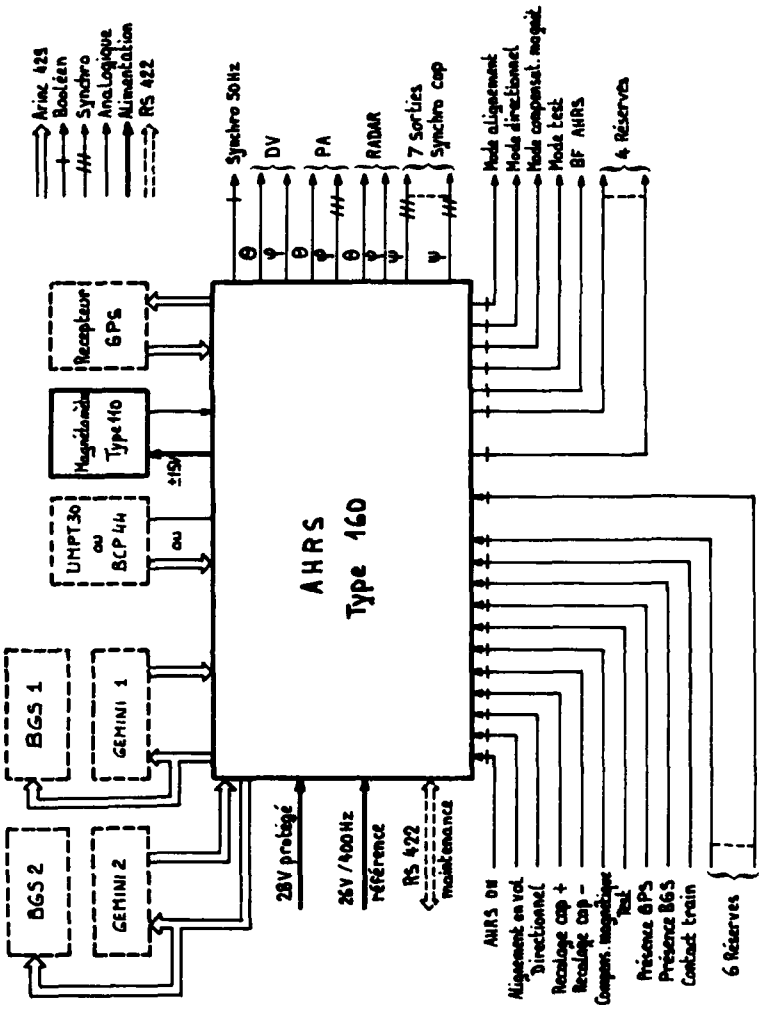



**FIGURE 3 : INTERFACES PREVUS POUR LE C 160 TRANSALL**

Enfin, de nouveaux besoins apparaissent, tels l'hybridation AHRS/GPS, la réalisation d'une fonction d'anémométrie basse vitesse pour hélicoptères, etc... Autant de modifications à prévoir qui viendront perturber la logique de la conception initiale.

Le système conçu doit être capable de toutes ces évolutions, mais surtout il doit demeurer compétitif sur les marchés internationaux.

La structure interne des équipements constitutifs du système a été spécifiquement étudiée pour satisfaire l'ensemble de ces exigences.

## 4. SOLUTIONS APPORTEES

### 4.1 Conception modulaire

L'organisation interne de la centrale de Références Primaires CIRUS, qui constitue le coeur du système, résulte d'une conception modulaire, ce qui lui confère la capacité d'évolutions et d'extensions fonctionnelles requises.

La centrale est organisée autour de trois cartes électroniques, chacune dotée d'une puissante unité de traitement, et qui fonctionnent de manière quasi indépendante l'une de l'autre :

- Une carte d'acquisition assure l'interface avec les données d'entrée du système : acquisiton et pré-traitement des capteurs internes ou externes, mise en forme des données numériques utilisées par l'unité de traitement centrale.

- Une carte unité centrale assure l'exécution des traitements opérationnels : paramètres air, plateforme virtuelle, AHRS.

- Une carte d'entrées sorties assure la réception des entrées spécifiques au porteur, et l'émission des sorties destinées aux différents systèmes utilisateurs.

Une telle structure permet une parfaite ségrégation fonctionnelle, ainsi qu'une réelle capacité à affronter les évolutions nécessaires au moindre coût. Ainsi, le changement d'un capteur ou d'une donnée d'entrée ne touchera que la carte d'acquisition, l'augmentation de la capacité d'entrées/sorties ne concernera que la carte d'interface correspondante.

De plus, une provision d'espace de deux cartes supplémentaires est prévue sur le fond de panier modulaire, ce qui accroît considérablement les capacités fonctionnelles et d'interfaçage du système, par exemple pour accueillir un récepteur GPS compact dans l'enveloppe même de la centrale.

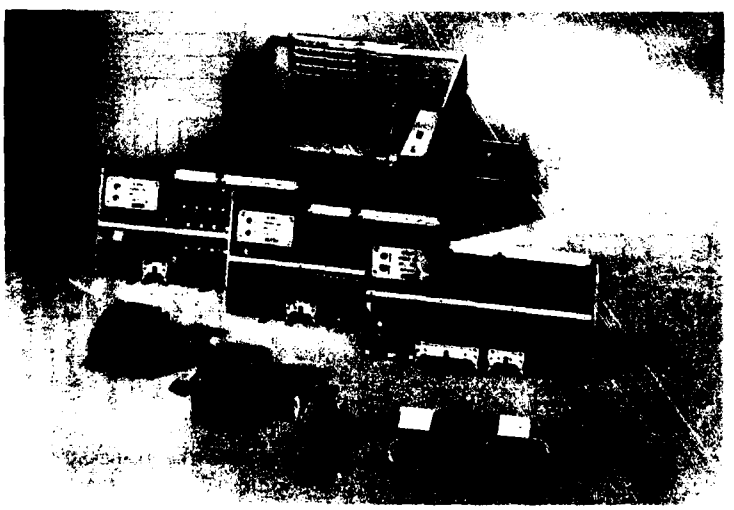


FIGURE 4 : CONCEPTION INTERNE DE LA CENTRALE DE REFERENCES PRIMAIRES

4.2 Qualité de la conception

Le développement du Systèmes de Références Primaires s'appuie sur une conception de haute qualité, dont les objectifs principaux sont d'assurer :

- Une grande fiabilité
- Une haute intégrité
- Une capacité à subir avec succès la certification prévue

**4.2.1 Fiabilité**

Le niveau de fiabilité requis est obtenu par le choix de composants matériels spécifiques. Parmi les plus significatifs, on peut citer :

- Les accéléromètres à pivot CROUZET 3152, dont plus de 5000 ont déjà été produits et installés sur différents types d'avions, de missiles et d'hélicoptères. Ces accéléromètres ont démontré, à l'usage, une excellente fiabilité.

- Les gyromètres accordés à paliers à gaz SMITHS DTG 2000. Par leur conception robuste et par le choix des technologies qu'ils utilisent, ces gyromètres (Réf. 3) se sont révélés les mieux adaptés sur le plan de la fiabilité. Ils ont été sélectionnés à l'issue d'un appel d'offres mondialement ouvert.

- Les capteurs de pression à membrane céramique CROUZET T 80, dont la conception simple permet d'atteindre à la fois des objectifs de coût réduit et de fiabilité élevée.

- Le magnétomètre statique triaxial CROUZET T 110 S3S (HSU).

- L'utilisation de composants électroniques intégrés, et en particulier, de composants spéciaux réalisés en technologie hybride, qui permet une meilleure tenue en température et un plus faible encombrement.

L'ensemble de ces éléments permet d'obtenir les MTBF suivants :

| | | |
|---|---|---|
| FDC | : | 3 000 h |
| PSU | : | 19 600 h |
| HSU | : | 58 800 h |
| Système SRP | : | 2 500 h |

*4.2.2 Intégrité*

Dans le cadre du système intégré de conduite du vol IFDS, des règles précises, pour la conception et le développement ont été mises en oeuvre afin d'assurer le niveau d'intégrité requis pour la certification.

On peut citer quelques une de ces règles au niveau du SRP :

- Système complètement redondé (duplex), sans aucune communication entre les deux chaînes redondantes pour éviter des points de panne communs.

- Alimentation de chaque chaîne par les deux réseaux de bord, ainsi que par une batterie tampon spécifique pour les coupures d'alimentation.

- Ségrégation physique des circuits redondants d'informations et de câblages (électriques, pneumatiques).

- Utilisation d'une source indépendante et simple (gyroscope de verticale) pour le lever de doute entre les deux chaînes duplex.

- Qualification complète suivant les normes civiles (DO 160 B) et militaires (AIR 7306, MIL STD 810 D - 461 C/462)

- Protection contre le foudroiement de tous les signaux critiques.

- Développement d'un logiciel de niveau 1 (fonction critique) conformément à la norme DO 178 A (voir § 4.4).

- Enfin, mise en oeuvre de procédures rigoureuses, impliquant une communication structurée entre l'Avionneur et l'Equipementier, pour la gestion de la configuration et des modifications au niveau documentation, matériel et logiciel.

4.3 Certification du système

**4.3.1 Bases de la certification**

Compte tenu des missions de transport en IMC de l'hélicoptère, on retient les minima opérationnels correspondant à la catégorie II.

- Visibilité horizontale : 400 mètres
- Plafond : 100 pieds
- Durée moyenne de la mission : 1 h

La certification DGAC, FAA et CAA du SUPER PUMA MK2 est prévue en 90/91 en vue des premières livraisons.

Les bases de la certification applicables au système sont :

- La FAR 29 Amendement 16 inclus
- Les critères de navigabilité IFR (lettre FAA du 15/12/78)
- Les conditions spéciales DGAC concernant le foudroiement

**4.3.2 Programme d'essais en vue de la certification**

*Ce programme d'essais se déroule en trois phases :*

a) Chez l'Equipementier

SEXTANT Avionique effectue des essais fonctionnels et d'environnement équipement par équipement, et vérifie l'intégration du sous-système SRP.

Ces essais mettent en oeuvre des moyens importants incluant une table d'essais inertiels, un banc de stimulation dynamique et des moyens importants d'essais en environnement (notamment EMC et foudre).

**FIGURE 5 : MOYENS D'ESSAIS INERTIELS**

b) Au banc d'intégration système Avionneur

Avec ce banc système, AEROSPATIALE vérifie le fonctionnement du système IFDS en dynamique avec les équipements réels SRP, PA et visualisations, et un simulateur dynamique de l'hélicoptère.

c) Essais sur hélicoptère

Le programme d'essais sur hélicoptère à Marignane se déroulera de mi-89 jusqu'à la certification de l'appareil.

Ces essais permettront de valider, dans tout le domaine de vol, l'ensemble références primaires, PA et interface équipage. Le fonctionnement en mode dégradé et après panne sera également évalué.

Par ailleurs, des essais spécifiques "foudre" seront effectués au CEAT (Centre d'Essais Aéronautiques de TOULOUSE) sur une cellule d'hélicoptère avec des maquettes.

**4.3.3 Analyse de pannes et de sécurité**

. En vue de la certification, SEXTANT Avionique a réalisé pour le SRP :

- une analyse de fiabilité pour chacun des équipements, jusqu'au niveau "composant fonctionnel"

- une analyse de panne (FMECA) incluant :

  . l'analyse des modes de pannes depuis les composants jusqu'au niveau sous-ensemble puis équipement

  . l'analyse des moyens de détection des pannes

  . le calcul des probabilités de pannes non détectées sur les paramètres de sortie (références primaires)

- A partir de ces éléments, AEROSPATIALE réalise, avec l'aide des Equipementiers concernés, une analyse de sécurité au niveau du système IFDS couvrant l'ensemble des événements redoutés. Cette analyse prend en compte les divers taux de panne, les moyens de détection, d'isolation et de reconfiguration, et les conditions d'emploi du système (maintenance, tests pré-vol, tests en vol, etc...).

4.4 Certification du logiciel

Il s'agit là, bien entendu, d'un aspect majeur en vue de la certification.

En accord avec l'Avionneur et la DGAC, le document de base est la recommandation RTCA-DO 178 A.

L'organisme chargé de la certification du logiciel est le CEAT de TOULOUSE, par délégation de la DGAC et du STTE (Service Technique des Télécommunications et des Equipements aéronautiques).

**4.4.1 Niveau de criticité**

Bien que certaines fonctions ne soient pas classées comme critiques, il a été décidé de réaliser tout le logiciel du SRP en niveau 1, devant la difficulté voire l'impossibilité de démontrer une ségrégation parfaite entre les logiciels de criticité différente et s'exécutant avec le même processeur.

D'autre part, on a fait le choix d'une même version du logiciel pour les deux chaînes redondantes pour les raisons suivantes :

- Coût prohibitif d'une vraie diversification du logiciel qui ne règle pas de toute façon les problèmes d'unicité pour la spécification en amont et la validation en aval

- Nécessité d'un logiciel de niveau 1 pour la certification (notamment FAA), même en cas de diversification logiciel.

**4.4.2 Méthodologie pour le développement du logiciel**

La méthodologie mise en place pour le logiciel SRP met en oeuvre les principes généraux désormais classiques pour les logiciels de haute intégrité, mais avec des contraintes très sévères de coût et de délai de développement (moins de 15 mois pour le SRP complet).

Les méthodes mises en oeuvre ont été les suivantes :

- Le découpage du processus de développement en phases chronologiques délimitant des travaux techniques cohérents

- L'attribution, à chaque étape de définition du logiciel, d'une activité de vérification dédiée

- La possibilité d'itérer certaines phases ou le cycle logiciel lui-même

- La mise en place et l'application de procédures et d'outils d'ingéniérie, de gestion de configuration et d'assurance qualité adaptés au projet et cohérents au niveau système IFDS

- l'obtention, à l'issue de chaque phase, de produits, (documents, logiciel) revus et maîtrisés suivant les procédures pré-établies par SEXTANT Avionique avec la participation d'AEROSPATIALE.

Au total, 9 revues ont eu lieu, dont 5 avec les Services Officiels, au cours desquelles ont été approuvés une vingtaine de documents réalisés au titre de la DO 178 A.

## CONCLUSIONS

L'ensemble des contraintes imposées pour le développement du système de Références Primaires ont abouti à la réalisation d'un système moderne utilisant les technologies permettant d'allier la performance à l'économie, et propre à satisfaire les besoins des avioniques des années 1990.

Une conception modulaire lui assure la capacité de s'adapter aisément à différentes configurations, et d'accroître ses fonctionnalités, sans remettre en cause le coeur de base certifié.

Les contraintes imposées pour satisfaire les exigences de fiabilité, de sécurité et d'intégrité dès la première applicaiton, en font un système répondant parfaitement à la notion de système critique pour le pilotage et la navigation des aéronefs civils et militaires.

## REFERENCES

Réf. 1 : "Modern strapdown system for helicopter"

J.L. ROCH. Fourteenth European Rotorcraft Forum. Milano, Sept. 1988.

Réf. 2 : "Main characteristics of an Integrated Flight and Display System for AS MK2 SUPER-PUMA"

B. RONTANI & S. RIOCHE. Fourteenth European Rotorcraft Forum. Milano, Sept. 1988.

Réf. 3 : "The design and development of a novel strapdown DTG incorporating a gas bearing and fabricated flexure hinge"

Dr. G. BEARDMORE. DGON Stuttgart, Sept. 1984.

| REPORT DOCUMENTATION PAGE | | | |
|---|---|---|---|
| **1. Recipient's Reference** | **2. Originator's Reference** | **3. Further Reference** | **4. Security Classification of Document** |
| | AGARD-CP-456 | ISBN 92-835-0552-2 | UNCLASSIFIED |

| | |
|---|---|
| **5. Originator** | Advisory Group for Aerospace Research and Development<br>North Atlantic Treaty Organization<br>7 rue Ancelle, 92200 Neuilly sur Seine, France |
| **6. Title** | FAULT TOLERANT DESIGN CONCEPTS FOR HIGHLY INTEGRATED FLIGHT CRITICAL GUIDANCE AND CONTROL SYSTEMS |
| **7. Presented at** | the Guidance and Control Panel 49th Symposium, held at the Ecole Nationale Supérieure de l'Aéronautique et de l'Espace in Toulouse, France, 10th—13th October 1989. |

| **8. Author(s)/Editor(s)** | **9. Date** |
|---|---|
| Various | March 1990 |
| **10. Author's/Editor's Address** | **11. Pages** |
| Various | 276 |

**12. Distribution Statement** This document is distributed in accordance with AGARD policies and regulations, which are outlined on the Outside Back Covers of all AGARD publications.

**13. Keywords/Descriptors**

| | |
|---|---|
| Guidance | Terrain avoidance |
| Control | Reconfigurable control |
| Fault tolerance system | Vehicle management |
| Software validation | Mission management |
| Terrain following | Maintenance diagnosis |

**14. Abstract**

This volume contains the 23 unclassified papers, including the Keynote address, presented at the Guidance and Control Panel Symposium, held at the Ecole Nationale Supérieure de l'Aéronautique et de l'Espace in Toulouse, France, 10th—13th October 1989.

The papers were presented covering the following headings:
— Trends in Integrated Flight Critical Systems;
— Advanced Fault Tolerant Design Concepts;
— System Architectures, Mechanization and Integration Issues;
— High Integrity Software Design Methodologies and Algorithms;
— System Validation, Simulation and Flight Test Experience.

| | | | |
|---|---|---|---|
| AGARD Conference Proceedings No.456<br>Advisory Group for Aerospace Research and Development, NATO<br>FAULT TOLERANT DESIGN CONCEPTS FOR HIGHLY INTEGRATED FLIGHT CRITICAL GUIDANCE AND CONTROL SYSTEMS<br>Published March 1990<br>276 pages<br><br>This volume contains the 23 unclassified papers, including the Keynote address, presented at the Guidance and Control Panel Symposium, held at the Ecole Nationale Supérieure de l'Aéronautique et de l'Espace in Toulouse, France, 10th–13th October 1989.<br><br>P.T.O. | AGARD-CP-456<br><br>Guidance<br>Control<br>Fault tolerance system<br>Software validation<br>Terrain following<br>Terrain avoidance<br>Reconfigurable control<br>Vehicle management<br>Mission management<br>Maintenance diagnosis | AGARD Conference Proceedings No.456<br>Advisory Group for Aerospace Research and Development, NATO<br>FAULT TOLERANT DESIGN CONCEPTS FOR HIGHLY INTEGRATED FLIGHT CRITICAL GUIDANCE AND CONTROL SYSTEMS<br>Published March 1990<br>276 pages<br><br>This volume contains the 23 unclassified papers, including the Keynote address, presented at the Guidance and Control Panel Symposium, held at the Ecole Nationale Supérieure de l'Aéronautique et de l'Espace in Toulouse, France, 10th–13th October 1989.<br><br>P.T.O. | AGARD-CP-456<br><br>Guidance<br>Control<br>Fault tolerance system<br>Software validation<br>Terrain following<br>Terrain avoidance<br>Reconfigurable control<br>Vehicle management<br>Mission management<br>Maintenance diagnosis |
| AGARD Conference Proceedings No.456<br>Advisory Group for Aerospace Research and Development, NATO<br>FAULT TOLERANT DESIGN CONCEPTS FOR HIGHLY INTEGRATED FLIGHT CRITICAL GUIDANCE AND CONTROL SYSTEMS<br>Published March 1990<br>276 pages<br><br>This volume contains the 23 unclassified papers, including the Keynote address, presented at the Guidance and Control Panel Symposium, held at the Ecole Nationale Supérieure de l'Aéronautique et de l'Espace in Toulouse, France, 10th–13th October 1989.<br><br>P.T.O. | AGARD-CP-456<br><br>Guidance<br>Control<br>Fault tolerance system<br>Software validation<br>Terrain following<br>Terrain avoidance<br>Reconfigurable control<br>Vehicle management<br>Mission management<br>Maintenance diagnosis | AGARD Conference Proceedings No.456<br>Advisory Group for Aerospace Research and Development, NATO<br>FAULT TOLERANT DESIGN CONCEPTS FOR HIGHLY INTEGRATED FLIGHT CRITICAL GUIDANCE AND CONTROL SYSTEMS<br>Published March 1990<br>276 pages<br><br>This volume contains the 23 unclassified papers, including the Keynote address, presented at the Guidance and Control Panel Symposium, held at the Ecole Nationale Supérieure de l'Aéronautique et de l'Espace in Toulouse, France, 10th–13th October 1989.<br><br>P.T.O. | AGARD-CP-456<br><br>Guidance<br>Control<br>Fault tolerance system<br>Software validation<br>Terrain following<br>Terrain avoidance<br>Reconfigurable control<br>Vehicle management<br>Mission management<br>Maintenance diagnosis |

The papers were presented covering the following headings:
– Trends in Integrated Flight Critical Systems;
– Advanced Fault Tolerant Design Concepts;
– System Architectures, Mechanization and Integration Issues;
– High Integrity Software Design Methodologies and Algorithms;
– System Validation, Simulation and Flight Test Experience.

ISBN 92-835-0552-2

The papers were presented covering the following headings:
– Trends in Integrated Flight Critical Systems;
– Advanced Fault Tolerant Design Concepts;
– System Architectures, Mechanization and Integration Issues;
– High Integrity Software Design Methodologies and Algorithms;
– System Validation, Simulation and Flight Test Experience.

ISBN 92-835-0552-2

The papers were presented covering the following headings:
– Trends in Integrated Flight Critical Systems;
– Advanced Fault Tolerant Design Concepts;
– System Architectures, Mechanization and Integration Issues;
– High Integrity Software Design Methodologies and Algorithms;
– System Validation, Simulation and Flight Test Experience.

ISBN 92-835-0552-2

The papers were presented covering the following headings:
– Trends in Integrated Flight Critical Systems;
– Advanced Fault Tolerant Design Concepts;
– System Architectures, Mechanization and Integration Issues;
– High Integrity Software Design Methodologies and Algorithms;
– System Validation, Simulation and Flight Test Experience.

ISBN 92-835-0552-2